Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Games with a Purpose - Improving 3D Model and Land Cover Data using Crowdsourcing

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Florian Felberbauer
Matrikelnummer 0625034

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Mitwirkung: Tobias Sturn, MSc.

Wien, 28.09.2012

(Unterschrift Verfasserin) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Games with a Purpose - Improving 3D Model and Land Cover Data using Crowdsourcing

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Florian Felberbauer
Registration Number 0625034

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Assistance: Tobias Sturn, MSc.

Vienna, 28.09.2012

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Florian Felberbauer
Dresdnerstraße 7/10/2
1200 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasserin)

Acknowledgements

First, I want to thank Tobias Sturn, my supervisor, for supporting me in elaborating ideas as well as for giving me tips or answering whatever questions asked. Supervising this thesis as well as all the review work to mostly uncivilized hours helped me a lot to make rapid progress.

I also want to thank Michael Wimmer for supervising the project and for giving feedback to various questions and problems.

Last but not least, I want to thank all my friends and colleagues who spend their free time helping me to collect data in my user study. I also owe special thanks to a friend that proof-read the whole thesis.

Abstract

A variety of 3D-model databases are available on the internet, but the process of finding the right models is often tiring. This is because the majority of the available models is barely annotated or the quality is low. Annotations often are ambiguous, vague or too specialized. Besides 3D-model annotations, remote sensing data can be ambiguous too. Global land cover maps like GlobCover, MODIS and GLC2000 show large differences in certain areas of the world. This lack of correct data is a problem, because it is a basic requirement for a variety of research areas and applications.

Consequently, this thesis aims at tackling both aforementioned problems. The task of recognizing and classifying images as well as 3D-models is easy to solve for human beings, but even today rather hard for computer systems. For that reason, this thesis makes use of the concepts of crowdsourcing. The quality of user annotations can be improved by collecting annotations from a variety of users and extract those with the highest frequency. To achieve this, a game has been implemented that unifies crowdsourcing and social games mechanics. This game consists of game-rounds which lead the user through the process of annotating 3D-models as well as land cover data. Also, a drawing round has been implemented to enable the user to classify a given land cover area using a pre-defined set of categories. As crowdsourcing is related to a large number of users, the focus is on implementing a game that provides incentives for users to spend their free time on playing, while solving useful tasks. To reach as many users as possible, the game has been implemented using only HTML5 and JavaScript to circumvent limitations due to missing plugins or external players and to support all systems, including mobile devices. It is also integrated into Facebook to further enlarge the number of reachable users.

The potential of the approach is demonstrated on the basis of a user study. The results show that the annotations with the highest frequency are good descriptors for the underlying 3D-models as well as for the land cover maps. None of the top annotations are incorrect for any model or map. Analyzing the user paintings also shows very good results. The majority of maps were classified correctly and even the distribution of categories over the maps are correct to a high degree. We thus show, that the combination of crowdsourcing and social games can improve land cover data and 3D-model annotations. These insights contribute to the ongoing "Landspotting" project, which is further explained in this thesis.

Kurzfassung

Eine Vielzahl von 3D-Modell-Datenbanken ist im Internet verfügbar, doch die Suche nach passenden Modellen ist oft ermüdend. Das liegt daran, dass die Mehrheit der verfügbaren Modelle entweder kaum annotiert sind oder die Qualität vorhandener Annotationen schlecht ist. Diese sind oft mehrdeutig oder vage oder im Gegensatz dazu zu speziell. Doch neben 3D-Modellen können auch Fernerkundungsdaten mehrdeutig sein. Globale Bodenkarten wie zum Beispiel GlobCover, MODIS oder GLC2000 zeigen starke Unterschiede an bestimmten Punkten der Erde. Das Fehlen von korrekten Daten ist ein Problem, da diese unabdingbar für eine Vielzahl von Forschungsrichtungen und Anwendungen sind.

Daher konzentriert sich diese Arbeit auf die Lösung der genannten Probleme. Menschen sind begabt im Erkennen und Klassifizieren von Bildern, aber auch von 3D-Modellen, während dies auch heutzutage noch schwierig für Computersysteme ist. Aus diesem Grund bedient sich diese Arbeit dem Konzept des Crowdsourcings. Die Qualität von User-Annotationen kann verbessert werden, indem diese von einer Vielzahl von Usern gesammelt und diejenigen mit der höchsten Anzahl an Übereinstimmungen extrahiert werden. Um dies zu erreichen wurde ein Spiel entwickelt, das Crowdsourcing und Elemente von Social Games miteinander verbindet. Das Spiel besteht aus mehreren Runden, die den User durch die Annotierung von 3D Modellen und Bodenkarten führen. Ebenso wurde ein Level entwickelt, in dem User gegebene Bodenkarten mittels vordefinierten Kategorien klassifizieren, indem Karten mit den passenden Kategorien bemalt werden. Da Crowdsourcing eine große Anzahl an Usern bedingt, liegt der Fokus auf der Enwicklung eines Spiels das den Usern Anreize bietet um ihre Freizeit mit spielen zu verbringen, während nützliche Aufgaben gelöst werden. Um möglichst viele User zu erreichen, wurde das Spiel ausschließlich mittels HTML5 und JavaScript entwickelt, um Limitierungen durch fehlende Plugins oder externe Player zu vermeiden und alle Systeme einschließlich mobiler Geräte zu unterstützen. Es wurde darüber hinaus in Facebook integriert, um noch mehr User zu erreichen.

Das Potential des Ansatzes wird anhand einer Userstudie demonstriert. Die Ergebnisse zeigen, dass jene Annotationen mit den höchsten Übereinstimmugen gute Deskriptoren für die zugrundeliegenden 3D-Modelle und Bodenkarten darstellen. Keine der bestgereihten Annotationen aller Modelle und Karten sind inkorrekt. Die Analyse der Daten die durch Userzeichnungen gewonnen wurden zeigen ebenfalls sehr gute Resultate. Der Großteil der Karten wurde korrekt klassifiziert und sogar die Verteilung der verschiedenen Arten von Bodenbedeckungen spiegelt sich in den Ergebnissen wider. Es wird weiterhin gezeigt, dass durch die Kombination von Crowdsourcing und Social Games Annotationen von Bodenkarten und 3D-Modellen verbessert werden kann. Die Erkenntnisse die durch diese Arbeit gewonnen haben tragen zum laufenden Projekt „Landspotting“ bei, das im Folgenden näher erläutert wird.

Contents

CHAPTER 1

Introduction

In the following, the motivation as well as the aim of this thesis is presented and the problems of existing approaches are explained briefly.

1.1 Motivation

Databases like Turbosquid, Google Warehouse or 3D Model Free are well-known resources for publically available 3D-models. These models are important for 3D-artists as well as computer graphics developers, because the creation of such models is a very time-consuming task and requires artistic skills. But finding models which fit the desired application is often also time consuming, because the majority of free and publically available 3D-models is usually either bad or not annotated at all. Annotations in general are strongly dependent on the user's perspective to a given context and time [32]. Because of this, given annotations are often ambiguous, volatile, vague or too specialized. Figure 1.1 shows four different models from Google Warehouse found for the keyword "tiger". Having meaningful annotations would therefore decrease the time dedicated to gathering models which match the demand. Additionally, the retrieval of 3D-models is an important part of multimedia information retrieval [39].

But next to 3D models, even remote sensing data can be ambiguous. Because research in the field of sensor technology has advanced significantly over the past decade, satellites are nowadays capable of recording land cover maps with a resolution of up to 300 by 300 meters. Global land cover maps like GlobCover [1], MODIS [28] and GLC-2000 [15] have been rendered in the last years. These maps evolved from remote sensing via satellites and contribute important information to land use, ecosystems and climate change. But the problem with these global land cover maps is that they show large differences in certain areas of the world. Figure 1.2 show how the same area is classified differently in three global land cover maps.

The size of the rectangles belongs to the resolution the satellites are capable of. MODIS classifies the area shown (blue rectangle) as "Non-Woody Savannahs", GLC-2000 (cyan rectangle) as "Cultivated and managed areas", while the red rectangle belonging to GlobCover is

Figure 1.1: Four models for keyword “tiger”, taken from Google Warehouse. Model four, which is the 41^{st} search result, is the first model of an animal tiger.

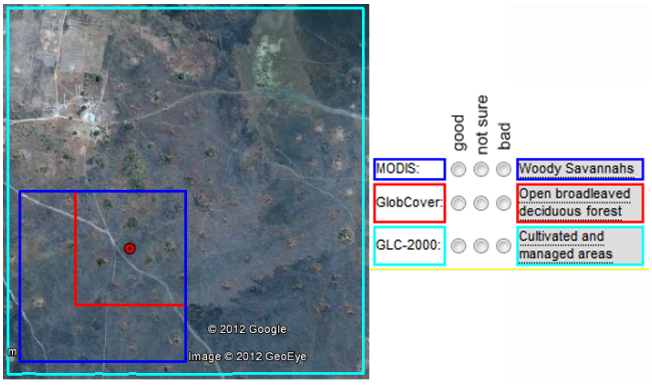


Figure 1.2: Classification of land cover by GLC-2000 (cyan), MODIS (blue) and GlobCover (red). This is a screenshot from the Geo-Wiki project’s website located at http://www.geo-wiki.org

classified as “Mosaic Cropland/grass or shrub or forest”. This demonstrates the aforementioned ambiguity of global land cover data. This lack of correct data is a problem for science, as it is vitally needed for a variety of applications and research areas. The determination of potential additional agricultural landuse [2] for ensuring food security or to grow bio fuel [12] is one imaginable application, but also analyzing climate and weather change by monitoring deforestation [5] or the collection of data for disaster response [16] are important applications where correct data is essential.

Therefore, the question is how to gather meaningful annotations. This thesis aims at collecting those annotations for 3D-models and especially land cover maps by creating a social crowdsourcing game where users provide annotations which are further compared and verified

to see if and how well land cover data can be improved by non-expert users.

1.2 Problem Statement

It has been stated in Section 1.1 that good annotations are vitally needed for a variety of research topics. Here, the current approaches for the extraction of annotations is described.

Automatic Approaches Algorithms for feature detection and -extration of images have been a research topic for several decades. But nevertheless, it is still a computationally hard and expensive task. Feature detection and extraction as a subfield of computer vision in general aims at local features of images like edges, corners, color distribution, blobs etc. Well-known examples for such algorithms are edge detectors like Canny, Sobel and Prewitt or scale-invariant approaches like SIFT. Content-based approaches are now making use of the aforementioned algorithms to search for images or 3D-models in large databases. By this, similar looking images or models can be found. For example, Google's image search is capable of searching for images which are similar to a given image. But the problem is that images with similar features do not necessarily match contentually. Figure 1.3 shows two images which have similar features from a computer vision point of view, but do not match on a semantic level. Different scale, rotation

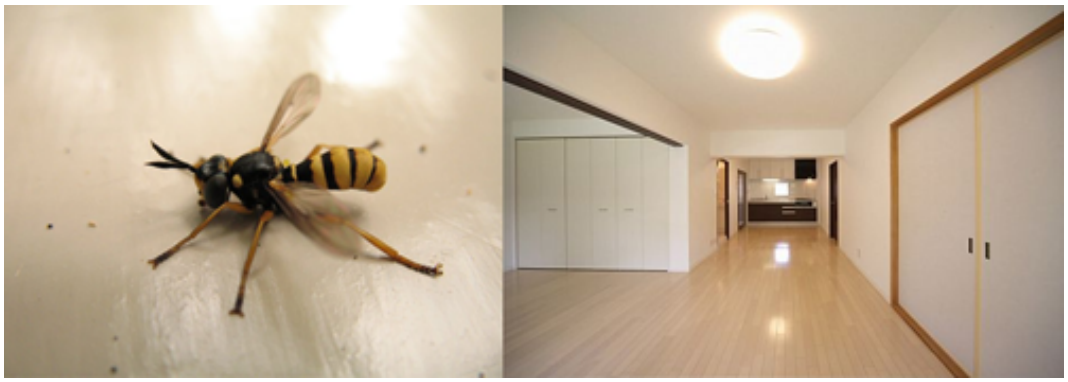


Figure 1.3: Image of a wasp on the left and the corresponding optical similar image found by Google image search. Wasp image "Erdwespe" from "Andi_und_Andi", source: www.piqs.de, Some rights reserved.

and viewpoint as well as the high variablity of a broad domain like the internet make this task even harder, especially for 3D-models [8].

Another, contrary approach for image retrieval is to use meta-data like captions, headings, filename or surrounding text on websites to extract images that match given search criterias. This, however, is not sufficient, as these meta-data are often rare, misleading or do not correlate to the given image [36]. Concept-based image indexing nevertheless is a more natural way for users to search for images or models wanted, for example by searching for "tiger" instead of specifying which features a tiger has. Another advantage is the possibility to provide more general search criteria. For example, searching for "mammal" could yield results like "dolphin"

or “cat”, although features like shape or color distribution are very different. For that reason, this thesis will direct it’s attention towards text-based search. Automatic algorithms have been developed to annotate 3d-models by analyzing shape as in [30] or by semantic correlation as in [39] as well as images, as in [38] and [26] to name a few. But automatic algorithms in general have two important drawbacks. First, the results achieved are often not satisfactory or need manual correction, often in the form of user relevance feedback. Second, they need a large set of well-annotated images or models to be trained. So manual annotation of images and models is needed for automatic algorithms, which forms a vicious cycle. For example, Ohbuchi and Kawamura [30] used an already annotated test set of about 900 models to train their algorithm. Annotating 900 models by hand is a very time consuming task that needs to be performed by experts having experience in the field of the application.

Manual Approaches It therefore seems natural to search for a way to get manual annotations in short time and for low costs. Practically, the recognition and annotation of images and models is a very fundamental task for the human brain and visual system. Therefore, a reasonable approach is to distribute the workload to a large base of users, so that each user only has to do a small part of processing. This approach of outsourcing work to a large user base is called crowdsourcing. The term crowdsourcing has been characterized by Jeff Howe in the 2006 article “The Rise of Crowdsourcing” from WIRED magazine. In his blog, he defines the term as: “Crowdsourcing is the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call.”

It is nowadays used by many projects like Wikipedia, Amazon Mechanical Turk, Foldit and others. Crowdsourcing overcomes the problems of automatic algorithms described above and also follows the trend in the field of mapping which came up in the last couple of years. While mapping has been performed by national institutions for centuries, there is a trend towards mapping by web users. That is because the highly risen number of internet users as well as the emergence of Web 2.0 lays the foundation for the involvement of a large number of users into such a process. Luis von Ahn did some spadework by channeling the human brain power into image annotation tasks by creating the currently very popular and well known ESP game [36]. This type of game is nowadays known as “Games with a Purpose (GWAP)” [35]. The idea of combining crowdsourcing with a game is very interesting, because people all over the world spend billions of hours each year playing computer games. Even if only a fraction of this time can be used to solve large-scale problems, one can imagine the huge potential of crowdsourcing in combination with computer games. As von Ahn stated in [36], all images indexed by Google could be annotated within 31 days if about 5000 people play a well-designed game for 24 hours a day. Even if this statement is from 2004 and the number of images indexed by Google increased significantly, the amount of potential players increased significantly too. The game is online since August 2003 and was played by 13,630 users in the first four months. While latest usage statistics are not available, the fact that the ESP game was acquired by Google and that the daily highscores show huge scores, shows that it is safe to assume that it is still very successful and popular.

Social networks like Facebook and Twitter became extremely popular in the last years and

millions of players participated in games like Farmville and Mafia Wars in those networks. Farmville, for example, reports 19,300,000 active users in July 2012 which tops the game "League of Legends", which Forbes Magazine elected to the most played PC game in the world. Even if it is unrealistic to compete with the top 3 global players in social games, these usage statistics show the potential of social games.

As a consequence to the aforementioned success of social games and crowdsourcing in general, the motivation of this thesis is to channel the abilities of the human brain to solve useful tasks by combining crowdsourcing and social games. But what are the requirements for a successful social game? First of all, humans require incentives to play a game and thereby become a task-solver. The most important incentive is entertainment. Paavilainen presented ten high-level heuristics for social games in [31] that defines what makes a social game entertaining. Following these heuristics listed in Table 1.1, the game has to be easily accessible, encourage playing in short, sporadic bursts, while the player is able to feel progress and to socialize within the game. As it is neither reasonable nor possible to fulfil every heuristic shown for every type of game, we chose those matching our game idea and implemented a social game to solve scientific tasks. The annotation of 3d-models as well as the annotation of land cover maps matches nicely with the concepts of crowdsourcing and social games, because the quality of annotations can be highly improved by involving more users in the annotation process [32]. Thus, the fuzziness of annotations can be reduced or even eliminated.

Spontaneity	Games should be easily accessible and understandable.
Interruptability	Infrequent and short game rounds shall be supported.
Continuity	Incentives for coming back to the game as well as rewards for every game round should be given and the players should feel progress due to a persistent, asynchronous game.
Discovery	The player should be able to collect achievements and discover new content regularly
Virality	By making use of virality like invite messages, new players can be attracted. Users should be honored for inviting other users.
Narrativity	In-game events should be described vividly and broadcasted to arouse curiosity.
Sharing	Reasons should be given to the user to share resources and informations.
Expression	Players should have an opportunity to express themselves through the game.
Sociability	Social contacts should be integrated into the game in form of goods and the communication with contacts should be honored.
Ranking	High score lists motivate for competition to play the game and to compete with friends.

Table 1.1: High-level heuristics for social games presented by Paavilainen[31]

1.3 Aim and Research Question

The aim of this work is the implementation of a collaborative but also competitive multiplayer computer game which exploits the potential of crowdsourcing to collect and verify user-based annotations for given 3D-models and land cover data as well as the categorization of areas of global land cover maps based on user paintings. It is going to consist of three game-rounds, each round tackling another problem. One round aims at collecting annotations for given 3D-models and global land cover maps. Another round enables users to choose between different brushes, each brush corresponding to a given category of land use. By brushing on a Google Map, users are able to flag the types of land cover shown on the map. Figure 1.4 shows an example of the classification round. All acquired keywords and land cover data are then stored. The game is based on an event-driven and thus non-blocking I/O JavaScript environment based on Google's V8 engine on the server side and on JavaScript and HTML5 on the client side, as this allows a browser-independent implementation. Furthermore, the game is integrated into Facebook. These decisions were made, in order to reach the biggest possible user base.

By collecting and analyzing this land cover data, the ongoing project "Landspotting" of the Institute of Computer Graphics and Algorithms [29] of the Vienna University of Technology in cooperation with other partners shall be supported. In the scope of this project, a web-based

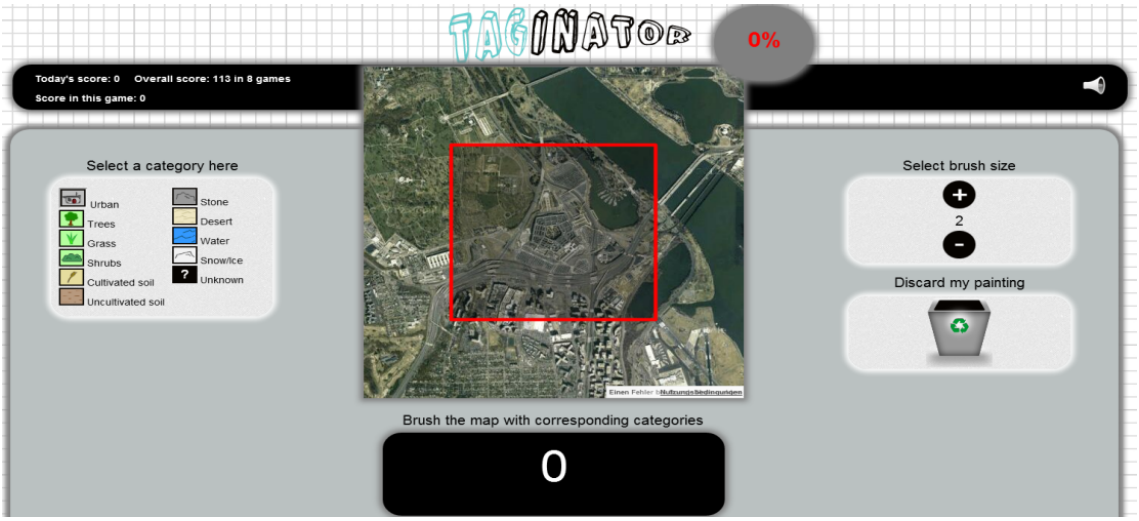


Figure 1.4: Categorization of global land cover maps by brushing maps with the given categories.

role play game is developed to refine global land cover maps. It is explained in more detail in Subsection 2.3.2.

This thesis' research question is whether user inputs are qualitatively valuable and if existent annotations can be improved. This includes the question if and how well land cover maps can be categorized or annotated, especially if the maps available have a low quality regarding to resolution, recognizability, distinguishability and confounding factors like clouds and fog. Likewise, the human ability to correctly annotate 3D-models and the quality of the resultion annotations is examined for models with different quality.

1.4 Contributions

The contributions of this thesis are:

1. Extension of the user annotation approach to 3D-models and land cover maps
   The idea of using crowdsourcing for annotation-tasks is extended to-3d-models and land cover maps by means of a social crowdsourcing game.

2. Extraction and verification of user annotations for global land cover maps
   On the basis of coordinates given by the Geo-Wiki project [12], user-annotations for land cover maps are collected and analyzed to examine if annotation is an appropriate tool to collect informations to refine the underlying maps.

3. Alternative method to generate land cover data
   The problem of different classifications of land cover maps by remote sensing is answered by collecting user drawings that identify and validate different land cover types for a given region.

4. Improved mechanics to reach more users
   By creating a game that is compatible with all platforms including mobile devices, and integration into Facebook, a large user base can be addressed to collect data from. HTML5 is used to create a rich user experience independent from plugins or external players. Additionally, models are pre-rendered into videos to support mobile devices which are not capable of WebGL yet.

5. Support for ongoing projects
   The ongoing project "Obfuscated Rendering" on the Institute of Computer Graphics and Algorithms [29] is supported, which aims at using obfuscated rendering techniques to prevent pattern recognition algorithms from detecting models automatically. The resulting videos can be used as CAPTCHAs and the user input can be compared to the annotations collected by our project. In addtion, this thesis does fundamental research to answer the question if global land cover data can be annotated and categorized correctly by untrained users to contribute to the Landspotting-project mentioned before.

1.5 Structure

This thesis is structured into the following chapters:

1. Introduction
   This very chapter.

2. Related work
   First, the state of the art in terms of crowdsourcing and user created content in general will be introduced. Thereafter, the state of the art in using crowdsourcing for generating image annotations is presented, as the concepts of these projects can be used for the annotation of global land cover maps and furthermore be extended to the field of 3d-model annotation. Subsequently, the latest projects in the field of collecting and improving land cover data using crowdsourcing are covered to gain an insight of the current work in this section.

3. TAGinator - a social crowdsourcing game
   The game implemented within the scope of this thesis is decribed in detail in this section. It starts with ideas and challenges incident to creating a multiplayer online game to collect data for both 3d-models and land cover maps while fulfilling the social game heuristics mentioned before. Also the challenges with platform independency, client/server synchronization and the problem of users playing at different times is covered. Subsequently, the user interface with all its elements as well as the gameplay itself and its game mechanics are described. At the end, the technologies used are described briefly as they enable cross-platform compatibility while focusing on performance.

4. Results
   In this chapter, the data extracted from the game is evaluated. This includes a small user study, where data is generated using a set of 3D-models and land cover data. The impact of certain game mechanics on the quality of the data is explored and the results are analyzed

in general and representatively based on three different 3D-models and land cover maps covering different qualities and resolutions.

5. Conclusion and Future Work
   This thesis will be concluded by a brief summary of the work including take-home-messages. Additionally, ideas and topics for future research are provided.

CHAPTER 2

Related Work

This section aims at presenting the state of the art in the field of crowdsourcing in general. After that, projects in image and 3D-model annotation are covered and finally, latest projects for collecting and improving land cover data are presented.

2.1 Crowdsourcing in General

It has already been stated, that the human brain is capable of solving specific computationally difficult problems like the recognition and annotation of objects and images and especially 3D structures. This computational power of the human brain can be bundled to solve scientific problems as well, which could potentially accelerate scientific progress. The big advantage of crowdsourcing is that users can gain expertise and contribute to solve scientific problems, even if the users are non-experts in solving a current problem or in the whole scientific discipline.

2.1.1 Foldit

A good example for such a scientific discovery game is Foldit. Foldit is a multiplayer online game in the field of biochemistry. Users are faced with the difficult problem of protein folding, wrapped in a puzzle-game. So the game aims at predicting protein structures, i.e. the determination of protein shapes for a sequence of amino acids. This prediction is essential because knowledge of the structure is the premise to gain insight into a protein's function [6]. In this work, biochemists post protein structures to the server for which the native structure needs to be found. Together with metadata and a parametrization, these structures form puzzles. For a specified time period, all users are able to download that puzzle and try to reach the highest possible score by reshaping. Native structures are compact and unique structures which are lowest in free energy. The scoring system is based on an energy function where the energy of a given structure is indirectly proportional to its distance to the native structure. Thus, high scores indicate good approximation of the native structure, which often includes very complex reshaping [6]. Figure 2.1 shows an example of such a folding. User scores are ranked against each other for the same

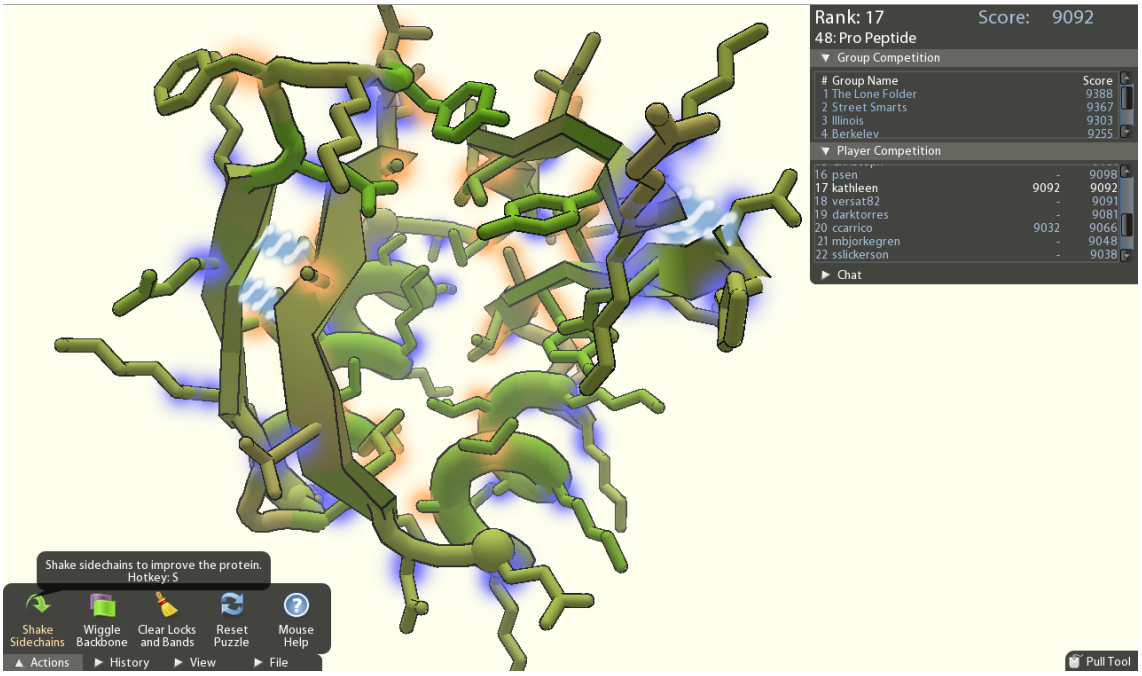


Figure 2.1: Foldit is a crowdsourcing game which aims at finding native structures for given protein structures. This is an important problem in biochemistry, which is hard to process automatically. This image has been taken from Foldit official website: http://fold.it/portal/info/science

puzzle and users can team up to solve puzzles together.

When a puzzle is closed, all solutions are collected and in turn analyzed by experts. This forms a feedback loop, as the analysis is used to improve the design of both the game and puzzles. The game design of Foldit is constrained by existing physical models and thus differs from the standard iterative approach of recurring design-, test- and evaluation stages. The results of the game have been evaluated by submitting the best user results for a couple of puzzles to the protein prediction competition CASP 2008. For half of the puzzles handed in, Foldit ranked inside top three. All of them had a better ranking than solely automatic algorithms, one puzzle even won first place. Even if Foldit is available as a desktop application and is hence not a typical social game on a social network, it is used to solve hard scientific problems by exploiting the power of human brains and empowers socializing by implementing team building and other social features.

Within the first two years after the initial release of the game in May 2008, more than 57,000 users have played the game, re-folding about 600 protein structures. A very impressive fact and a good example for the potential of crowdsourcing is that the crystal structure of M-PMV, which is an AIDS-causing monkey virus, has been accurately approximated by Foldit-users within ten days [24]. This has been an unsolved problem for scientists for the last 15 years.

2.1.2 How Do Humans Sketch Objects?

Sketching is a generic form of communication and has been used by humans since pre-historic times. Even today, the only possible rendering technique most people are capable of is sketching. Eitz et al. consequently investigated in [9] how people sketch and how well they perform in recognizing of sketches. Additionally, they developed a computational recognition system for sketches and compared human and computational performance. The innovation here is that contrary to earlier research, they do not assume that sketches represent their real-life counterparts in "some well-engineered feature space" [9]. This assumption does not hold anyways, because of lacking artistic skills of most humans, which are substituted with iconic, simplified or oversubscribed representations of objects.

For training data, they collected 20,000 sketches from 1,350 unique users. This huge amount of sketches has been collected by crowdsourcing, using the platform Amazon Mechanical Turk (AMT) that is described in Subsection 2.1.3. The sketches correspond to a set of 250 predefined categories. These categories were chosen to cover everyday life objects that are recognizable from shape without the need of further context, but are specific enough to avoid a large number of subcategories. Analysis of the gathered sketches shows a median number of 13 strokes per sketch, while the stroke length decreases over time. The authors consequently suggest that humans follow a coarse-to-fine approach for sketching. [9]. To investigate the human performance on sketch recognition, another user study on AMT has been performed, where users were asked to assign the best fitting category to sketches. This user study shows an average of 73.1% correctly assigned sketches, while the results vary strongly for different categories [9]. These variations are due to semantical or geometrical similarity of sketches from different categories.

To compare these results to computational sketch recognition, a robust classifier has been built, representing sketches as frequency histogram of visual words that are gathered using a visual vocabulary based on k-means clustering of feature vectors. A more detailed description of the classifier would go beyond the scope of this section, so interested users may be referred to [9], Sections 5-7. The classifier identified 56% of the sketches correctly, compared to 73.1% success rate for humans. But the authors claim that 73.1% is a lower bound because AMT users try to complete the task as fast as possible instead of as good as possible. So even if there is room for improvement, further research on this topic could lead to new areas of application. For example, human-computer interaction by the use of sketches could help illiterates to access available informations on the internet. [9]. A video demonstrating the framework can be found at the website of the University of Brown [34].

2.1.3 Amazon Mechanical Turk

Another interesting development in the field of crowdsourcing is Amazon Mechanical Turk (MTurk). It is an internet marketplace for micro-tasks called "Human Intelligence Tasks" (HIT). They vary massively from text and audio annotation to translation and recognition tasks, transcriptions, verification tasks, assignments like company to address mapping and many others. Every HIT is rewarded by a specific amount of money, which varies from about 61 US-Dollar for a 2.5 hour voicemail transcription to 0.01 dollar for receiving emails, for example. Additionally, there are some tasks free of charge. The amounts given are a spot test and are subject to varia-

tion. 61 US-Dollar for a 2.5 hours HIT is by far an exception of the rule, because HIT typically require very little time and the rewards stay within the limits of a couple of cents. Adapting this system to scientific needs to collect useful data is however challenging. As there is no a-priori knowledge about correct or wrong answers in general, users could provide nonsense-answers to increase their income by decreasing the time needed to complete a task. Kittur et al. show two experiments in their work [25], where they had MTurk users rate 14 Wikipedia articles. These ratings have been compared to Wikipedia administrators subsequently to determine the quality of the ratings.

In the first experiment, 210 ratings from 58 users were collected within 48 hours. About 48% of the given answers to a specific question were useless and about a fourth of all tasks have been completed in less than a minute. Further analysis of the ratings showed that only 8 out of 58 participating users were responsible for about 73% of all invalid responses. So instead of a wide user base trying to cheat, only a small group tried multiple times. But besides cheaters, the correlation between user ratings and administrator ratings was statistically negligible. Invalid ratings are not rewarded, but the identification costs time and thus money. In their second experiment, Kittur et al. changed the rating task so that the effort for giving bad or invalid ratings is the same than for giving good ones. They introduced some control questions with corresponding defined answers to exclude invalid participants. This highly improved the quality of the ratings given by MTurk users. Only 7 out of 277 compared to 122 out of 210 ratings were invalid and the correlations between user ratings and administrator ratings also increased significantly. This leads to the conclusion that Amazon Mechanical Turk is a good platform for very fast crowdsourcing for low overall costs, but it has to be treated with caution as the lure of "making a fast buck" is high. Thus, the need of control questions to avoid exploitation of the system is high.

2.1.4 reCAPTCHA

The third state of the art project covered here is presumably the best known one, since it is used by almost everyone online on a daily basis. It is called reCAPTCHA and was developed by von Ahn et al. [37]. The abbreviation CAPTCHA stands for "Completely Automated Public Touring test to tell Computers and Humans Apart" and is a very popular security mechanism on the web to prevent bots from abusing online services like e-mail providers, social networks, wikis, ticket sellers, blogs and many more [37]. The CAPTCHA approach asks the user to decipher an image consisting of distorted characters, which is hard for computers to do. By estimations from von Ahn et al. from 2008, humans around the world solve over 100 million CAPTCHAs every day. They claim that these ten thousands of hours are wasted with the pure CAPTCHA approach. Because of that, the reCAPTCHA approach is now channeling this effort into something useful. It goes one step further by asking the user to decipher two different images with distorted characters, where one is a reference image with a known ground truth and the other is unknown. The unknown images typically are scans from books, which have not been correctly recognized by Optical Character Recognition (OCR) software. To avoid gaming of the system, users do not know which one of the images is the reference image. Additionally, both the reference and the unknown image are distorted to avoid automatic algorithms to pass. If the user input for the reference image is correct, the input for the unknown image is assumed to be correct as well. Figure 2.2 shows, how a reCaptcha is generated. The authors of [37]

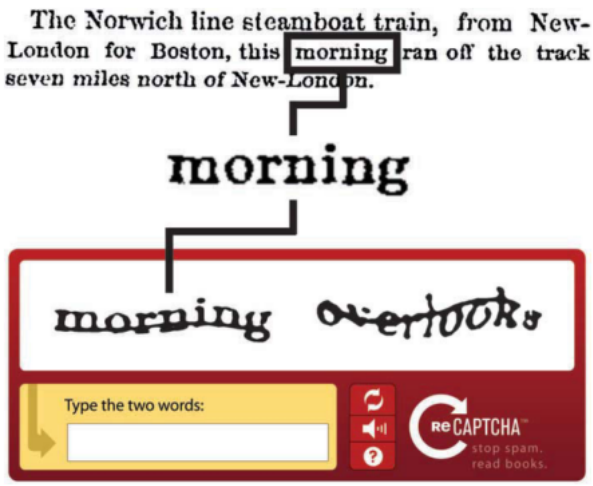


Figure 2.2: reCAPTCHA channels human power of recognizing distorted characters for digitizing books and other documents. This image has been taken from [37].

report that the accuracy of user-recognized words exceeds 99% which matches the quality of professional transcribers. By the time the paper was published, more than 40,000 websites used reCAPTCHA. In 2009, Google took over the project to improve their Google Books Library Project.

2.1.5 Related Examples

Two general examples for crowdsourcing or public-resource computing are SETI@home and Wikipedia. SETI@home is a project dedicated to the "Search of Extraterrestrial Intelligence" (SETI) and uses millions of computers in homes and offices all around the world. The project collects huge amounts of data from a secondary antenna of the world's largest and most sensitive radio telescope operated by Cornell University in Puerto Rico [3]. This data is then distributed among all participating computers, where each computer analyzes a small part of the data. The results are then sent back to the server and further processed. As a more detailed explanation of this project would go beyond the scope of this section as it is not directly related to crowdsourcing, it can be said that the SETI@home project was the first project showing the potentials of public-resource computing. In 2002, they already had 3.91 million users whose computers processed 221 million work loads with $1.873 * 10^{21}$ floating point operations [3].

Wikipedia is a multilingual, web-based platform of free content with the goal to build an encyclopedia. The word "Wikipedia" is built-up from the Hawaiian word "wiki", which means "quick" and the word "encyclopedia". The articles and entries are written collaboratively and emerge from an extremly large base of anonymous internet volunteers. Any user can edit articles to improve them or to fix errors or mistakes, except for articles which are restricted from editing to prevent misuse. The user base has the potential to consist of all people having access to the internet. Thus, a large variety of ages, educational and cultural backgrounds are represented. Users contribute in form of articles, entries or media, which are then further reviewed to ensure quality

standards with regards to copyright violations, controversial allegations, personal opinions or non-verifiable sources or references. Wikipedia has means of version control, so that mistakes or vandalism can be easiliy reverted and a high number of editors make sure that every edit of an article is an improvement to the former version. One big advantage of Wikipedia compared to printed Encyclopediae is that content can be added or improved in real time, while articles, for example about recent events, take years to appear in printed Encyclopediae. Wikipedia has grown extremely fast, with about 500,000,000 unique visitors per month. The english version of Wikipedia counts about 4 million articles with a bullish tendency. Furthermore, articles exist in about 280 different languages.

2.2 Image Annotation

In this subsection, the state of the art in the field of image annotation is presented, as this is the scaffolding for the annotation of 3D-models.

2.2.1 ESP Game

The very popular and successful ESP Game developed by Luis von Ahn at Carnegie Mellon University is a trailblazer in crowdsourcing and the development of games with a purpose. The field of application for games with a purpose is eclectic and ranges from security and computer vision to internet accessibility and internet search, just to name a few [35]. Many important applications on the world wide web require sets of meaningful image annotations, for example to improve algorithms for search engines or tools helping visually impaired people. Search engines typically analyze text adjacent to images and assume that this text is correlated to the image. But this assumption is often wrong. Thus, the only possibility is to label manually, which is very costly. The ESP Game therefore aims at collecting image annotations through a simple web-based game, where two players type keywords for an image shown. In case the image is not recognizable or the users don't agree on a keyword, there is a button to pass the image. So if both players decided to pass the current image or they typed in the same word, another image is shown. They do not have to type the same word at the same time, but all inputs for both users are matched against each other. If there is a matching keyword, both users get rewarded. This continues for at most 15 images or until the game clock expires. If the users agreed on all 15 images, they both get recompensed for.

Additionally, some words are so called "taboo words" [35], which are words that a specified number of other users already agreed on. With these taboo words, a wider range of keywords can be collected. The more taboo words there are, the higher the score the users get when agreeing on a keyword. If a specific number of taboo words is reached, this word is marked as done and will not be presented to other users anymore. To avoid cheating by communicating with each other, the players are paired randomly so that nobody knows who the other player is. A couple of anti-cheating strategies have been implemented. It could be possible for example that many users agree on a strategy to type in the same word over and over again. If this behaviour is detected, a massive number of bots is inserted into the gameplay so that the chance of being paired with a bot is high. Due to this approach and the fact that the pairing is random among

Figure 2.3: The popular ESP Game from Luis von Ahn, which was taken on by Google in 2006. This image is a screenshot from the ESP game, copyright holder is GWAP.

hundreds or thousands of players, the chance to cheat successfully is very low [36]. There is also a mechanism which transforms words that have been typed very often into taboo words for the entire game session to avoid the strategies mentioned. Even if the strategie is like typing "one" for the first image shown, "two" for the second and so on, this is not a big problem because of the threshold for good image annotations. To become a problem, the same image would need to be presented in the same chronology to a large number of teams, each team having agreed on the same cheating strategy. As the game design changed significantly since initial deployment, Figure 2.3 shows how the game looks like in July 2012. The ESP is online since 2003 and is still very successful in 2012. Within four months after the initial deployment, a total of 13,630 people played the game, generating 1,271,451 annotations for 293,760 images [36] with some players playing more than 1000 games or 50 hours. Google has taken on ESP game in 2006 and there are unfortunately no current usage statistics available. But highscores and user rankings indicate that the community is still very active.

2.2.2 KissKissBan

KissKissBan is a competitive human computation game for image annotation developed by Ho et al. [17]. The game is different from others as it unifies collaborative and competitive gameplay into one game. There are three players involved simultaneously. Two form the couple and one is the blocker. Like in the ESP game, the couple tries to agree on an image annotation. In contrast, the blocker's goal is to prevent them from agreeing on annotations. In the ESP game, players are rewarded for entering matching words. The problem with this approach is that players enter common words, since the chance for a match increases. This problem cannot be solved entirely

with taboo words, because they bias the player's behaviour. For example if the word "man" is blocked because it is a taboo word, then players may enter "guy" instead [17]. As opposed to the ESP game's taboo words, the words entered by the blocker in KissKissBan are not visible to the couple. Because the couple would get a penalty for guessing blocked words, they are prompted to enter more diverse annotations. One advantage is that the role of the blocker forms a natural anti-cheat mechanism, because if the couple agrees on the same words all the time, the blocker is going to block them the next round. The gameplay allows to enter blocking words for seven seconds in the beginning. After the seven seconds have expired, the couple has 30 seconds to agree on image annotations. Every time a player from the couple enters a blocked word, the time remaining will be reduced by five seconds. If the couple nevertheless agrees on an annotation within the time limit, it wins the game. Otherwise, the blocker is the winner of the game round.

To make it fair, the roles are switched every five rounds, as the game consists of 15 rounds [17]. One other advantage of the competitive approach of KissKissBan is that common or easy words can be collected because usually the blocker provides them and more diverse words are provided by the couple, as they do not want to get blocked. A blocking word is not collected before a player types the same word and is blocked. Because players can get blocked repeatedly within a game round, it is possible to collect more than one annotation at once, which increases the efficiency compared to the ESP game. Unfortunately, no such impressive usage statistics as for other works presented here exist, because the authors only did a small user survey with 17 players, generating 5521 labels in 537 games. For evaluation purposes, the authors re-implemented the ESP game without taboo words and generated 11.54 distinct labels per image compared to 6.56 for the re-implemented ESP game. As taboo words are missing and the sample size is small, the significance of the comparison is anyone's guess. The main idea to generate more diverse annotations by having a blocker which prevents the players from agreeing on the easiest words possible could however be beneficial for future projects.

2.2.3 TagCAPTCHA

An interesting extension to the aforementioned CAPTCHA approach is called TagCaptcha and was developed by Morrison et al. [27]. This approach also takes advantage of the human ability to recognize and classify images, but instead of asking users to decipher distorted text, they have to describe images shown. As the approach uses both a reference set and unknown images, it could be considered as a variation of the already mentioned reCAPTCHA approach. Users are shown different images, where some being from a well known reference set and the other images being unknown, not yet annotated ones. If two users agree on a keyword for one of the unknown images, it is added to a set of pending annotations and further added to the verification set if other users agree on the same keyword too. To make it even harder for automated programs to attack, only parts of the images are shown, while the user can see the image in full size by simply mouse-over. The approach uses a two-step matching strategy to avoid false negative ratings as there is an inherent subjectivity involved when annotating images with English free-text words [27]. The first step is the comparison of the plain given user annotations. If they do not match, the words are compared using WordNet and a specific similarity measure, called the WUP distance [27]. This WordNet soft match has the advantage that semantically close

annotations can be accepted. If an image from a dolphin is shown for example , then using the WUP distance, the words dolphin and mammal may have a high similarity.

Choosing a good similarity threshold is however a tradeoff between usability and security. By security, the vulnerability to bot attacks is meant, which in turn depends on parameters like the size of the image database, the vocabulary size and the numbers of verification images shown to the user. But it is still possible to train algorithms on the reference set, as the approach is designed to use publically available images. Following the common approach of distorting images while keeping the semantic informatin intact, this problem can be reduced or eliminated. In a small user study performed by the authors of [27], 12 participants achieved a success rate of 70%, which is significantly lower than the stated success rate of 90% for the CAPTCHA approach. This is due to the fact that labeling images is a much more subjective tasks than entering text shown in an image. Another factor which is lowering the success rate are language barriers, for example users with a mother tongue other than English may have difficulties finding the right word for the image shown. But even spelling mistakes are a problem, which have been reduced by providing a spelling helper to the interface in the test system of [27]. Considering these factors, an overall success rate of 70% seems more than suitable and could be further improved by adding a translation system to reduce problems caused by language barriers.

2.2.4 Draw Something

Draw Something is a commercial, social drawing game for smartphones. The aim of the game is that users draw pictures corresponding to given words. It can be downloaded for free in the Apple App Store as well as in the Google Play Store and is available in 13 different languages by now. It is possible to play collaboratively with friends or competitively with foreign people. The user interface provides the functionality to add friends to a list, where games can then be started. The first player draws a picture to a given word, which can be selected from three possibilities with different difficulty levels. After the drawing has been finished, the picture as well as the drawing process is then stored and the second player is notified about the available drawing via push notification. The second player can then watch the process of drawing and has to enter the correct word by choosing letters from a given set. If the guess was correct, both players are rewarded with gold coins. Words which are more difficult, are rewarded with more coins. These gold coins can be used to buy additional color palettes, to extend the set of colors available. Then the roles change and the second player draws a picture, while the first one has to guess. After each guess or drawing, a short message can be added, which is then shown to the other player.

As long as the drawings are guessed correctly, a counter beside the friends name in the list is incremented. The counter is reset, if a user passes because the drawing is too hard to recognize. Occasionally, bombs are given to the users, which can be used to buy special words or to get help for guessing the drawings. Like in most other commercial apps, items like gold coins or bombs can be bought ingame for real money. Figure 2.4 shows an ingame screenshot of the Android version of the game. Draw Something is very popular and successful. The game was downloaded 37 million times and is ranked first in Apple's App Store. Draw Something was developed by OMGPOP, but the game as well as the whole company has been sold to Zynga, the world's leading provider of social games, for $200 million. But according to Forbes [11], since Zynga took over the game, the number of users decreases by about five million each month.

Figure 2.4: The popular Draw Something app. This screenshot was taken from the Android version of the game. Copyright holder is Zynga Inc.

2.3 Land Cover Data

As the last section, the current state of the art in the field of crowdsourcing geographical informations is presented. This is a relatively new operational area, which becomes more and more popular these days.

2.3.1 Geo-Wiki project

The Geo-Wiki project [12] was developed to tackle the problem of missing accurate global land cover maps. As already mentioned in Chapter 1, global land cover maps show large differences, which is a problem for many research areas and fields of application. The project follows the concept of crowdsourcing, as this enables almost every internet user, including non remote sensing experts, to help to improve and validate these maps. In this project, the global land cover maps MODIS, GLC-200 and GlobCover are used to analyze and resolve differences. GlobCover has a resolution of 300m x 300m, which is the best among the three named land cover maps, but the resolution is still not good enough to distinguish land cover features accurately, especially for non-expert users [12]. Thus, the project takes advantage of the Google Earth platform, as the resolution of the maps provided is up to 50cm x 50cm in contrast. Through a Web Map Service (WMS), users are able to visualize each of the three global land cover maps introduced as well as the disagreements among two or the overall disagreements among all disagreement maps with a varying threshold.

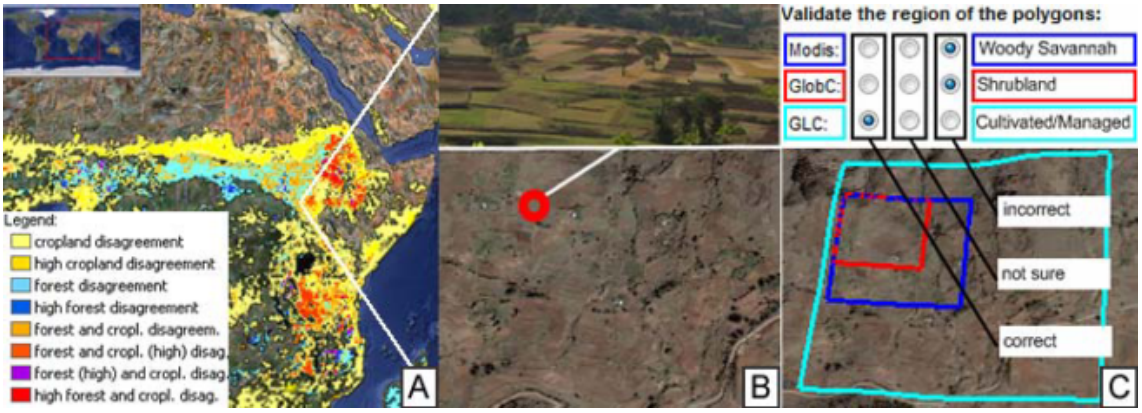


Figure 2.5: The Geo-Wiki project aims at validating global land cover maps with crowdsourcing. For a certain area, users are presented classification derived from existing land cover maps (C), disagreement maps (A) as well as augmentation in form of geo-tagged user photographies to get an insight into the real land cover present at the given position (B). The image is taken from [12].

These disagreements have been recorded for two key land cover classes, namely cropland and forest. In addition, the implementation shows features like overlaying global land cover maps with planimetric data like national borders, towns and roads [12] or geo-tagged user photographies. These features offer additional valuable informations about the landscape and the land cover for a specific area. With all the mentioned features, users are now able to validate land cover either on a pixel level or for a whole area at once by deciding if the land cover shown is correct, incorrect or not sure. Figure 2.5 shows the calculated disagreement maps as well as additionally the augmentation of land cover maps with geo-tagged user photographies. Having user-validated land cover does not redeem necessary approaches of accuracy assessment of land cover, but can contribute to an improvement. The authors of [12] claim that involving people outside the scientific community is still a challenge which could be tackled by unifying the discussed approach with competitive elements from computer games and social networks to reach the maximum user base. Another challenge is to meet the requirements for accuracy, i.e. to make sure that the system is not vulnerable to attacks and misuse. The authors state that this could be accomplished by following the concepts of Wikipedia, where articles are reviewed by volunteers who can dismiss or complete them.

2.3.2 Landspotting

The project “Landspotting” [2] is a spin-off of the Geo-Wiki project and is coordinated by the International Institute of Applied System Analysis (IIASA). It is an international, scientific non-governmental institute which aims at researching problems like climate change and is funded by scientific institutions from all over the world. A couple of partners are involved and the Vienna University of Technology is one of them. The Institute of Computer Graphics and Algorithms is responsible for the game environment design and implementation. It includes a review of

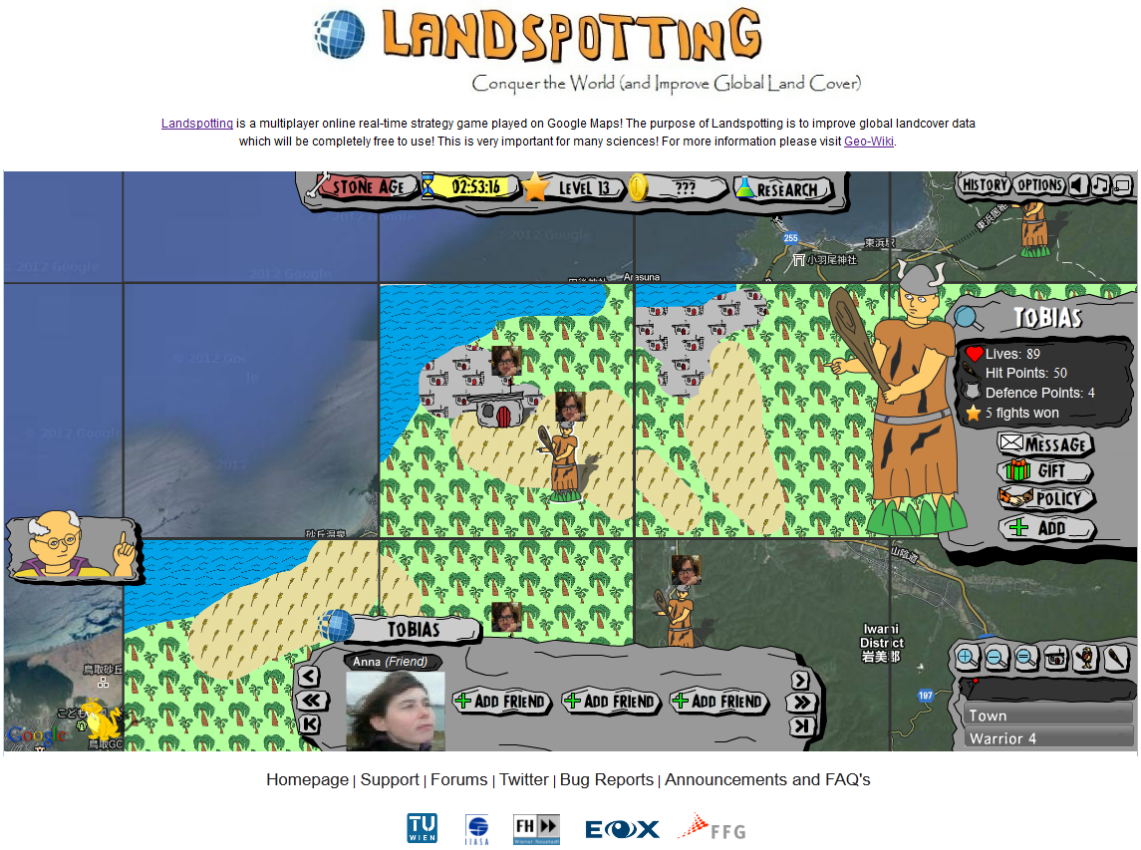


Figure 2.6: The landspotting game is the result of the Landspotting project to improve global land cover maps. This screenshot has been taken from the game available at apps.facebook.com/landspotting

existing online games as well as social and video games which have a wide user base. Following the successful concepts of existing games, a prototype was created within the Geo-Wiki environment. The game aims at improving the quality of land cover information by providing an incentive to the users to keep them playing. This goal is pursued by implementing a game based on Adobe Flash which has similar game mechanics as the well-known game “Civilization” and is integrated into Facebook. This includes diplomacy, research of a tech tree, warfare and the obligatory acquisition of resources through harvesting etc. Additionally, social mechanics are supported like purchasing gifts, communication, post messages and others. The information and validation gained from the game are then used by the Geo-Wiki system to improve data quality. A screenshot of the game is shown in Figure 2.6.

2.3.3 Crowdsourcing geographic information for disaster response

M. Goodchild and J.Glennon analyzed the need and potential of volunteered geographic information (VGI) in case of disaster response [16]. The Wenchun earthquake in 2009 as well as Hurricane Katrina in 2005 and the disastrous Indian Ocean Tsunami in 2004 are good examples for disasters, which caused a lot of damage and a high number of injuries and deaths. While agencies analyzing geographic data are more or less overchallenged in case of emergencies, mainly because of staff or money shortage, the collective of internet users could manage a huge amount of workload. One good example for the potentials of VGI is the wildfire in Santa Barabara, California in 2008. Because of disadvantageous wind conditions, the fire spread extremely rapid. Just in near-realtime, text reports, photographs and videos about the fire were published on the internet and citizens from Santa Barbara discovered that in-time informations are available rather from these sources, including websites from local newspapers, community groups or other services than from official agencies. Several volunteers additionally synthesized these information in form of maps based on services such as Google Maps, which formed sources that were easily accessible and oftentimes more up to date than official sources. But the quality of data gathered by non-expert users are likewise a problem for this kind of application and further research is needed in the formalization of rules to check geographic information against its context. The example of the Santa Barbara wildfire however showed that there is a huge potential for this field of application.

2.4 Technologies Used for our Project

Cross-platform and cross-browser compatiblity is a strong requirement for the game implemented within the scope of this thesis. Further on, interactivity on the client side is important, as page reloads are not acceptable for the game design. Another important aspect is the scalability, because the number of potential users is huge, as the number of active players for the ESP game or games like Farmville shows. But besides scalability also performance is essential when handling a large number of users. These requirements led to the decision to implement the server architecture using Google's V8 engine as well as Websockets and other frameworks described below.

2.4.1 Google V8 Engine

Google's V8 engine is a high-performance, open source JavaScript engine written in C++, which is used in Google's browser Chrome [21]. It implements ECMAScript as specified in ECMA-262 and runs on Linux, MacOS and Windows. The V8 engine enables applications written in C++ to expose objects and functions to JavaScript code and can be run standalone or embedded into C++ applications. It further compiles and executes JavaScript code and offers garbage collection and memory allocation. The V8 engine is significantly faster than other Javascript engines like JScript (Internet Explorer) or SpiderMonkey (Firefox). This can be seen in the results of two benchmark suites ran on Firefox v.15, Chrome v.21 and Internet Explorer 9. Both benchmarks were run three times per browser. The average results are shown in Table 2.1. The improvement in speed depends on the amount of code executed and increases by the number of

Mozilla's Dromaeo Benchmark Suite - All JavaScript Tests		
Mozilla Firefox v.15	**Google Chrome v.21**	**Internet Explorer 9**
375.83 runs/s	695.493 runs/s	74.996 runs/s

Google's V8 Benchmark Suite - version 7		
Mozilla Firefox v.15	**Google Chrome v.21**	**Internet Explorer 9**
4096 points	6787.33 points	122.33 points

Table 2.1: Average benchmark results, performed on an Intel Core2 6300 @ 1.86GHz with 3.00 GB RAM on Windows 7 Professional 32 Bit

times functions are executed. One key feature of V8's performance is fast property access, which describes a methodology to access JavaScript properties. While most engines use a dictionary-like data structure for storage of properties, V8 create hidden classes. Whenever a property is added, the hidden class changes and a transition from the preceeding to the succeeding hidden class is stored. This seems inefficient in the beginning, but it enables reusing the hidden classes and no dictionary lookup is needed in the future. Further on, the use of hidden classes enables the engine to use inline caching. This means that on initial execution, the hidden class of an object is determined and the property access is optimized by predicting that the class will also be used for same objects which appear in future code. If the prediction is false, the inline cache is patched.

Another key feature of performance is efficient garbage collection. V8 stops the exection of programs when a garbage collection cycle is performed ("stop-the-world"), while only a part of the object heap is processed in each cycle to minimize the consequence of stopping the application. The heap is divided into a new space, where objects are created and an old space, where objects are moved which survived a garbage collection cycle ("generational") [19]. Further details about the V8 core concepts can be found on the Google Developer website [20].

2.4.2 Node.js

Node.js is a server-side JavaScript environment based on the aforementioned Google's V8 engine, written mainly in C++ and JavaScript [22] and distributed under the MIT license. Its focus lies on low memory consumption and high performance. The server architecture used for the game implemented within the scope of this thesis is built with Node.js, as it fits the needs described in Chapter 4 well. While most environments like Apache are based on multithreading for concurrency, Node is based on a non-blocking, asynchronous I/O eventing model [7, 33]. This is explained with a simple example of a database request, which in procedural code is written as:

```
var result = db.query("SELECT * FROM table");
//results are available here
```

This means that the application has to wait until the query finished and the result is returned, which is wasting CPU cycles. So multithreading aims at running other threads, while one is waiting for the database results and is thus blocked. On multicore systems, each core executes a different thread in parallel, while on single core systems, threads are executed consecutively. This requires context switching, which is expensive and consumes memory due to execution stacks. But even on multicore systems, multithreading is hard to implement and burdened with problems like deadlocks and errors due to protection of shared resources, which are difficult to debug. Additionally, because the operating system decides which thread to execute and for how long, developers are losing control to a certain degree [33].

Event-driven programming can solve these problems by design, because it relies on event notifications. This means that the application registers for certain events and is then notified through callback functions when the operation has completed. But I/O operations for single threaded event-loops have to be non-blocking. Otherwise the advantages of the event-driven architecture would be nullified, because waiting for an I/O operation to finish would block the whole thread. This requirement is the main reason why event loops are not very popular, because most libraries do not support non-blocking I/O. This includes database libraries like *libmysql_client* as well as asynchronous DNS resolution [7]. Additionally, callbacks are difficult in many languages, for example C, because of missing support of closures or anonymous functions. This is where JavaScript comes in. It was designed to be used with an event loop, because it supports anonymous functions and closures and handles I/O operations through callbacks. In comparison to the synchronous database request above, the asynchronous, non-blocking way using an anonymous callback function would be:

```
db.query("SELECT * FROM table", function(result) {
  //results are available here
});
```

Node.js follows a strict approach of asynchronicity to provide a purely event-driven, non-blocking way to develop highly concurrent applications. Thus, to retrieve information from hard drive, network, database or other processes, a callback function must be passed to the function call. For example, a TCP server in Node emits a *connection* event each time a client connects and an HTTP upload emits a *body* event for every packet received [7]. Thereby, a function never blocks for I/O operations, the control is returned to the application right after the function call and the callback is then called when data is ready. This architecture is good at handling a high number of small, dynamic requests, like most web applications are designed for. Node forces developers to follow the asynchronous model from the beginning, while asynchronicity is one of many options to be used in other environments.

Node enjoys great popularity since two years ago and the community is growing fast. This may be because the rise of HTML5 reduces the need to use alternative client-side frameworks. Thus, developers have to become familiar with the concepts of JavaScript to create rich user experiences. With Node, developers can now use the same language for both server-side and client-side code. While the documentation is not mature yet, it is very easy to get started. The code for setting up a simple HTTP server which responds to each request with “Hello World” is shown below:

```
var http = require("http");
http.createServer(function (req,res) {
  res.sendHeader(200,
    {"Content-Type": "text/plain"});
  res.sendBody("Hello World\r\n");
  res.finish();
}).listen(8000);
```

A large number of modules have been developed by the community, which can be easily installed using the shipped package manager *npm*. These modules include full-featured web application frameworks like Express, asynchronous interfaces to relational- or NoSQL databases or websocket implementations like Socket.io. Express and Socket.io are introduced below. A list of modules developed for Node.js can be found at https://github.com/joyent/node/wiki/modules.

2.4.3 Express.js

Express is a fast and minimalistic web development framework for Node. It aims at performance and on providing small and robust tools for HTTP servers [10]. It is built on Connect, which in turn is an extensible HTTP server framework for Node, that supports high performance middleware. Connect comes with more than 20 often needed middlewares like a request parser, session support, a cookie parser and others. Express further broadens the available functionality. This includes HTTP helpers for caching or redirects and a robust routing system. It also supports environment based configurations as well as over 14 different template engines, including Jade, Haml, EJS and Haml-Coffe for CoffeeScript support [13]. Using Express, one can only use what is needed, without being forced to use specific libraries. The framework follows the Model-View-Controller (MVC) pattern by offering functionality to expose objects, functions and modules to client-side scripts. With that, it is possible to strictly separate code from design by developing client-side HTML code and pass data to it from the server-side. Another useful feature is the ability to load configurations from a key/value store like Redis or Memcache. This way, the resources that are commonly used for the game can be efficiently cached, including especially the session storage needed for user authentication. The following example shows, how a simplified server including routes can be started.

```
var express = require("express");
var app = express();

app.get("/", function(request, response){
  response.render("views/index", {
    title: "myTitle",
    userlist: myUsers
  });
});

app.listen(3000);
```

This example shows how efficient code can be by making use of Node.js and Express.js. The call to *require()* loads the dependencies from hard disk and *app.get('/')* registers a route. So every GET-request to the route *'/'* will be handled by the code above, but a POST-request for example would not. Besides the URI, a function is passed to *app.get()* which provides access to the request and response objects. For example, it is possible to access the session object by *request.session* and by using the response object, redirects can be done or a resource can be rendered. For example, if the user requests the URI of the start page, highscores and achievements can be passed, depending on the user-id stored in the session. In the example above, the view *index.jade* is rendered and an object is passed to the client-side, containing arbitrary data. Because it is possible to configure the default templating engine and the public directory globally, the file ending *.jade* can be omitted as well as the full path to the *views* folder.

As Node is built in form of middlewares, it is also possible to pass an additional middleware-call as an argument to any route. This is very useful to execute other code before the main code inside the route is executed. An example is the authentication process to check if the user is already logged in into Facebook. For every page served, first the authentication object is retrieved and it is checked if a Facebook-id is passed. The processing happens in ticks, which means that every called middleware has to call *next()* to pass control to the succeeding middleware. If for example the authentication fails because of a missing id, the middleware simply redirects the user to a login page and *return*s. Because of the missing call to *next()*, the execution stops and the content is protected.

2.4.4 Socket.IO

Communication between server and client is essential for every network application. This communication is mostly handled over the HTTP protocol, where the client sends a request to the server, which then serves data in response. This means that a new request needs to be sent whenever new data is needed. For standard web applications, this basically entails a page refresh. When a user fills out a form and submits it to the server, for example, the input is checked on the server-side and the results are then sent back to the client. Dynamic applications that require client-server communication at frequent intervals however have other requirements. A simple example for that is a chat application, where new messages shall be displayed without the need for the user to refresh the page. This is basically achieved by an underlying API which sends requests to the server and makes the results available to the developer, who can then manipulate the DOM to show new messages. For our purpose, communication between client and server without page refresh is essential, as client and server exchange data continously during the game. This includes, for example, updates of the game countdown as well as synchronized triggering of loading game- and score screens.

The best known technology is probably AJAX, which is an apronym for “Asynchronous JavaScript and XML”. It uses the XMLHttpRequest object that is implemented in all modern browsers. The WebSocket API however is a slightly different approach, which uses TCP sockets over the unsecure *ws* or the secure *wss* protocol. The main difference is that the server as well as the client can push messages to each other, eliminating the need of client-side polling to check

for new data available. Using the WebSocket API, developers can handle bi-directional client-server communication without the overhead of AJAX requests, as long as the browser supports it [40].

If WebSocket is not supported by the browser, fallbacks can be used. One example is Flash, which provides a simple alternative. The obvious drawback is that Flash is not installed on all clients, including Apple iPhone or iPad. Another fallback is AJAX long-polling, which however is not optimized for sending messages. In spite of that, developers still have to implement detection of supported technologies, fallback transports, event handling and interfaces to the server-side solutions. With Socket.IO, this is no longer necessary, as it simplifies the WebSocket API and uses feature detection to decide which technology can be used to establish a connection. Further on, Socket.IO unifies the APIs of supported fallback transports. Both a client and server side implementation is available, which makes development faster and easier. Besides Node.js, also ports for Android, Java, C++, Perl, Phyton and others exist [14]. We use Socket.IO for all communication between clients and server in the game. Whenever a user connects, a socket is registered and further used to send and receive messages. These messages include updates of the game timers, notifications when a client finished loading various resources, the reception of entered data, the transmission of scores or just notifications to the user.

Using Socket.IO with Express can be achieved through handshake/authorization mechanism. This enables user-defined functions to be invoked on a new websocket connection. This function will be called before the connection is completed, so connections can be accepted or rejected and HTTP headers can be accessed. This is necessary to access the cookie, which stores the sessionID of the Express session. Express uses a session store for user sessions which is a MemoryStore by default, but even Redis or other key/value stores can be used. In the handshake function it is now possible to load and manipulate the session corresponding to the user request. The following, simplified code snippet shows how to use Socket.IO with Express to bi-directionally communicate between client and server and manipulate the user session [14].

```
var sio = require("socket.io"),
  express = require("express"),
  app = express.createServer(),
  MemoryStore = express.session.MemoryStore,
  parseCookie = require('connect').utils.parseCookie,
  sessionStore = new MemoryStore(),
  Session = require('connect').middleware.session.Session;

app.use(express.bodyParser());
app.use(express.cookieParser());

//Setup session store
app.use(express.session(
  {
    store: sessionStore,
    secret: "0123456ACF81D",
    key: "express.sid",
```

```
  }
));

//pass Express instance to Socket.IO
io = sio.listen(app);

io.set("authorization", function(handshake, callback) {
  if(handshake.headers.cookie) {
    handshake.cookie =
      parseCookie(handshake.headers.cookie);
    handshake.sessionID =
      handshake.cookie["express.sid"];
    handshake.sessionStore = sessionStore;

    //get session from session store
    sessionStore.get(handshake.sessionID,
       function (error, session) {
        handshake.session = new Session(handshake, session);
        callback(null, true);
      }
    });
 }
});

io.sockets.on("connection", function(socket) {
  //Session object is available here
  var sessionObject = socket.handshake.session;

  //listen for message "messageFromClient"
  socket.on("messageFromClient", function(data) {

    //send message "responseToClient" to the client
    socket.emit("responseToClient", otherData);
  });
});
```

Error checking is omitted in this example for the sake of readability and of course on the client side, Socket.io needs to be configured. Anyways, one can see that the whole setup of Socket.IO using Express, including receiving and sending messages from and to clients can be done in a few lines of code.

CHAPTER 3

TAGinator - a social crowdsourcing game

The game "TAGinator" is available at https://apps.facebook.com/taginator and is explained in detail in this chapter, covering the game mechanics and their motivations as well as the user interface and user interactions.

3.1 Game Mechanics

During the last couple of years a trend towards crowdsourcing was visible and the potential of it is impressive. Additionally, social games on social networks like Facebook or Twitter are extremely successful. As already mentioned in Section 1.1, games like Farmville or Mafia Wars have millions of active users per month. Therefore, the idea was to combine the principles of crowdsourcing with the huge user base reachable by social platforms.

Right from the beginning we wanted to implement a multiplayer game which is fun to play and competitive, but also supports collaborative gameplay and motivates the player to solve scientific tasks while enjoying the participation. This need for client-server synchronisation paired with the requirement to show 3D-models as well as land cover maps and the limited resources of web browsers led to the decision to implement a round-based game. That way, loading times can be hidden from the user and lags due to connection problems or performance issues are not as critical as for other genres. Additionally, the workload for the client is less than it would be for a 3D game, which is important for mobile devices and also older machines.

The game aims at collecting annotations for both 3D-models as well as for land cover maps and evaluating the quality and correctness. Additionally, information about the different types of land cover are collected by providing a way for users to paint on Google Maps. A screenshot for the drawing round can be seen in Figure 3.1. For each game up to 5 players are randomly paired together until a countdown has expired. The game then starts and shows one of three different

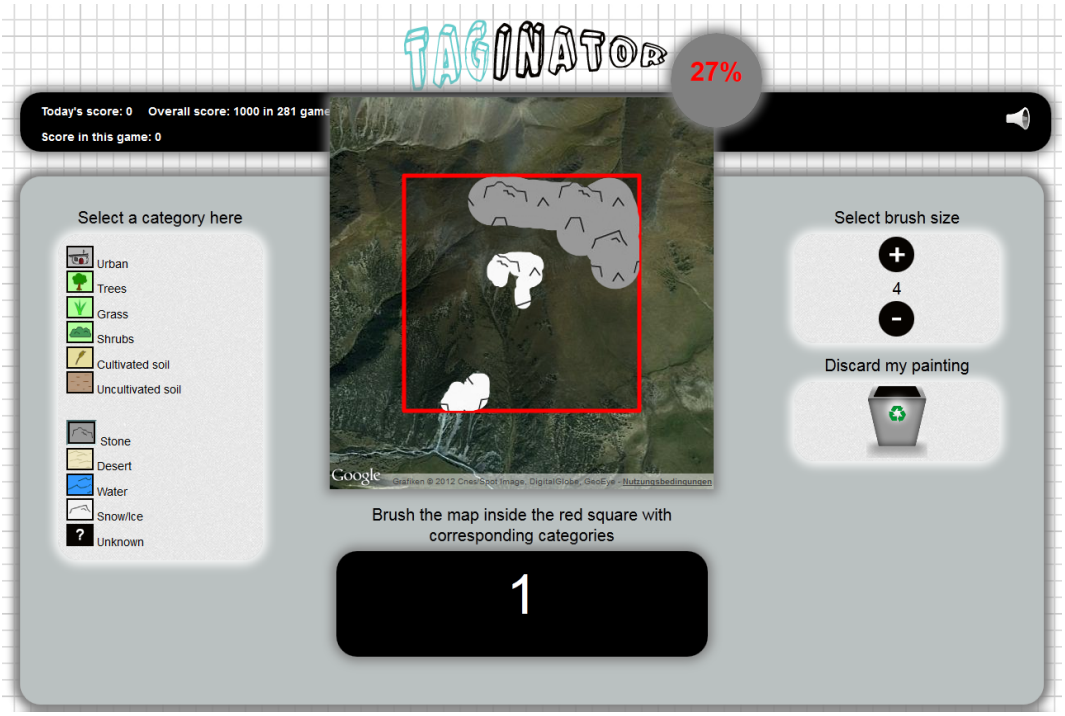


Figure 3.1: Screenshot of the drawing game-round of TAGinator

game screens to the user. These are explained in the following. Detailed descriptions of the user interface for each game-round including screenshots can be found in Section 3.3.

3.1.1 Annotation of 3D-Models

3D-models are vitally needed by developers and artists working on 3D-applications. Annotations for publically available 3D-models are however often bad or the models are not annotated at all. The availability of well annotated models would decrease the amount of time needed to create or find models which fit the needs of those developers. This is explained in more detail in Section 1.1. We therefore implemented a game-round to annotate 3D-models. The round shows a video of a 3D-model in the middle of the screen. The video loops endlessly and the model is rotating around its up-axis. Users have to enter textual annotations that match the model shown. Each user is allowed to enter up to five distinct annotations until the game countdown has expired. We decided for 5 annotations because we do not want to limit to only 1 annotation as in the ESP game described in Subsection 2.2.1. This way we can collect as many annotations per player as he or she is able to provide within a given time. We do not force players to enter five annotations in every round, so providing less than five is also fine.

However, once the first player entered the maximum of five distinct annotations the countdown speeds and turns red and the finished player is marked in the user interface to indicate which player was the first to finish. The faster countdown is an element to amplify the competitive character of the game and motivates players to provide more annotations. Once the

countdown expires or all players entered five annotations the game round is finished and a score-screen is loaded. It shows the player's scores for each entered annotation. The scores depend on the number of matches of annotations among all players. As a result, the more players entered the same annotation the more points they get. By doing this the quality and correctness of annotations can be evaluated by assuming that annotations with a high number of occurrences tend to be correct. Additionally, each user is rewarded with a bonus score that is calculated by the number of times an annotation for the given model or map has been entered by other players before. Besides player's scores, a list of annotations is shown on the score-screen. These annotations can be positively or negatively rated by players to further improve the quality. This is explained in Subsection 3.1.4.

3.1.2 Annotation of Land Cover Maps

Meaningful annotations are not only important for 3D-models but also for remote sensing. Global land cover maps like MODIS, GLC-2000 or GlobCover have been rendered in the last couple of years, but they show large differences in many areas of the world. As correct land cover data is strongly needed, for example, for monitoring climate change or for ensuring food security we collect annotations to help to improve those areas of disagreement. This is explained in more detail in Section 1.1.

The game-round designed for collecting user-annotations for land cover maps basically has the same mechanics as for the 3D-model-annotation round described before. Of course, a land cover map is shown instead of a 3D-model for which users have to enter annotations. The map is centered at coordinates given in Latitude and Longitude provided by the Geo-Wiki project described in Subsection 2.3.1. Those coordinates define regions of large disagreement among different land cover maps. To show the region of interest within a map a red rectangle is shown which covers an area of $4km^2$. We have chosen this size as it has been tried and tested to be halfway between too detailed and too small. When showing larger areas to the user, the image gets too small to recognize important features of the underlying land cover. Smaller areas would require a higher zoom-level to still fit into the game layout, but for many of the interesting areas only maps with lower resolution exist. Figure 3.2 shows such a map in its highest available resolution, which has been proven to be already hard to annotate. The area of $4km^2$ does not directly correspond to the resolutions of satellites like MODIS or GLC-2000. But one main goal of this thesis is fundamental research to answer the question if and how well land cover maps can be annotated and classified by untrained users to contribute to the Landspotting-project. So we chose this area as it is the same size as used in the Landspotting-project and thus allows to compare the gathered information.

3.1.3 Land Cover Categorization by User Drawings

Besides textual annotations, we additionally want to generate information about the land cover and its distribution over the map in form of user drawings. Hence, the user is able to categorize land cover maps by painting on a map. In the middle of the screen the map is shown like in the annotation-round for land cover maps described before.

Figure 3.2: Land cover map with low resolution which has been proven to be hard to annotate.

The Geo-Wiki project aims at letting users verify or falsify the automatically generated land cover categorization of global land cover maps in areas of disagreement. MODIS, GlobCover and GLC-2000 define between 17 and 23 categories which also differ among each other. This is not applicable for our needs as the categories defined are too many, too special and the distinction between the categories is not an easy task for non-expert users. For that reason we agreed on 11 distinct categories which cover the most important land cover types. These categories are available in the game as icons as shown in Figure 3.3. Having these 11 categories, users are able

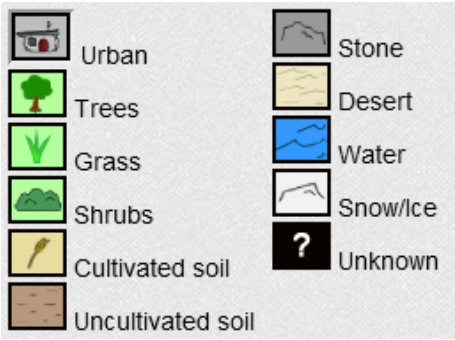


Figure 3.3: Categories available as brushes for drawing on a map.

to easily distinguish between categories and to choose the one matching the land cover shown on the map.

By choosing the right category, players are able to assign these categories to the types of land cover shown on the map. Different brush sizes are provided to support painting of coarse and fine structures on the map. As for the game-rounds before, a countdown indicates the time left for the round. There is more time available compared to the annotation rounds and the countdown does not speed once the first player finished the map. We do not want to hurry players while

drawing as we expect that the quality of the painted image decreases when the available time is too less. The percentage of already filled pixels is shown above the map and turns from red to green once a player filled enough pixels of the map to be valid for further processing. If not enough pixels were filled, the map is discarded as we do not expect to get reasonable results from those drawings. When the countdown has expired, a score-screen is loaded that shows the player's scores. The scores depend on the similarity of the painted maps among all users except those who did not fill enough pixels of the map.

3.1.4 User Ratings

After the annotation game-rounds, each player sees his own entered annotations as well as the number of matches and the resulting points for each annotation in a table. Additionally, scores for other players are shown in the same table, but without the entered annotations. In another table, all annotations from other players are shown, except those entered by the current player, to avoid up- or downrating his or her own annotations. Each player can now up- or downrate good or bad annotations two times each by clicking the icons next to the annotations. The rating is then double-checked with the database and scores are given if the rating matches the ratings already stored in the database. This approach shall ensure the quality of annotations, because each player takes on the role of a referee to some degree. If the rating does not match the ratings already stored, no score is given. We decided not to penalize players who enter annotations which are then rated bad. Likewise players entering annotations which are later rated good are not rewarded.

The intention behind this decision is that the rating feature could be overused, especially for bad ratings. If a user gets penalized for a bad rating, then scores could be influenced intentionally to manipulate the player's final score to be tactically prevented from winning. By only rewarding ratings that match the overall opinion about an annotation, no influence on other user's scores is possible. If a user enters nonsense intentionally, other users have the ability to downrate, which helps to separate good from bad annotations later on. Player ratings are not shown to other players to avoid that they influence each other in the decision which annotations match the given land cover map or 3D-model and which do not. There is no user-rating after the game-round where users have to paint on a given map because of the lack of qualitative features which can be judged by players.

3.1.5 Scoring

As the game idea is based on agreements of multiple players on the same annotations, it is neccessary to analyze the player annotations and drawings. Thus after each game round, the clients send all entered data to the server. For annotation rounds, the data consist of the annotations, as well as the time taken to enter them. All annotations are then added to a data structure, with annotations as the key and an array of user objects as the value, so that the number of agreements can be retrieved efficiently. This data structure is then traversed and each annotation for each player is stored to the database. The essential score thus depends on the number of agreements for each annotation. The higher the number of agreements is, the higher the score will be. For annotations without agreements, no score is awarded. All annotations with more than one

agreement are then further processed, by calculating a bonus score. This is done by fetching the number of distinct users from the database, which submitted the same annotation for the same model or map too, excluding the player itself. Thus, a bonus score is calculated for each player separately, which may result in different bonus scores for different players. As the number of equal annotations will rise as more players play the game, the upper bound of bonus scores is set to three. The annotations as well as the essential score, the bonus score and a list of annotations to be rated are then assigned to each player and the data is sent back to the client, which then shows these data as shown in Figure 3.10.

For the drawing round, the score is calculated by comparing the percentage of filled pixels for each category among all players. Drawings where the percentage of filled pixels is below a certain threshold will be excluded from comparison and get zero points. The remaining drawings are then further analyzed and an average per category is built. This average per category is then compared to the percentage of filled pixels for each player. The minimum of the average and the user drawing per category is taken and summed up for each category to calculate the score. This way, the more similar the drawings of different players are, the higher the score will be. The scores of all players are then finally sent to the client and shown to the player.

For the asynchronous single player mode, where bots are inserted to simulate multiplayer behaviour, the scoring strategy is basically the same. The only difference is that annotations and drawings are fetched from the database right after the game starts. For pioneer mode, the first user to play the pioneer game will be awarded with one point for each annotation entered as well as five points for the drawing. When a second player finished the pioneer game, the data of both players are then compared and the score is calculated like mentioned before. The difference of the temporary score for the first player and the final score after the second player finished is then stored to the database and the first player is notified about the gain or loss of points when entering the game the next time.

3.1.6 Socializing and Gamifications

Following the high-level heuristics for social games from Paavilainen [31] presented in Section 1.1, we need to provide further incentives for players to play the game regularly and to fulfill the requirements of today's social games. As heuristics like "Spontaneity", "Interruptability" and "Continuity" were already realized through the game design itself we decided to implement the heuristics "Discovery", "Sociability" and "Ranking" too. The player's scores are presented per game round, per day and overall. By having separate scores it is possible to present high score lists for the all-time bests, monthly bests or even daily bests. That way the possiblity to climb the ladder and to compete with other players by achieving higher scores always becomes the focus of attention and motivates players to spend their time in game. Beyond that, achievements can be collected as a further motivation and players can post their scores on their Facebook wall as well as invite friends to play the game. Highscore lists are shown on the starting page as well as achievements already collected and those still receivable. As described in Subsection 3.1.8, communication channels like chats were omitted intentionally and the names of fellow players are not shown during the game to prevent users from trivially agreeing on cheating strategies. To still fulfill the requirements of the heuristic of sociability, users are able to post messages on their Facebook wall or invite friends to play the game. Further on, at the very end of the game,

when the final scoring round is finished, the names of the fellow players are shown to amplify the competitive character of the game.

3.1.7 Handling Single Players

The game is designed and intended to be played by a large number of users, as we expect to get the best results from having a maximum of five players available for every game round. But as a social game needs to enable playing in short, sporadic bursts and to be accessible without barriers, we decided to implement an asychronous mode for the case that not enough players are online to start a game round. This is expected to become important especially in the time after the initial deployment because of the low level of awareness. So whenever a user joins the game, he or she is paired to an open, pending game round. If no game round is pending at the moment, a new game is created and a countdown is started to wait for other users to join. But without an asynchronous mode, the game could not start if no other players join the game before the countdown expires. This is due to the fact that achieving a score would be impossible, because scoring is based on the agreement of multiple users on the same annotations or on matching drawings for land cover classification. So there is the need to provide the same game experience to single players as for playing with other human players.

The idea consequently was to replace missing players with emulated users without letting the player know, because the social aspects would suffer from knowing that the user is playing on his or her own. Whenever the countdown expires and only one player is assigned to a game, the asynchronous mode becomes active, emulating a random number between one and four other players to lead the singe player to believe in playing a regular game with other humans. This is achieved by fetching pre-recorded annotations and land cover classifications from prior game rounds from the database. For the annotation round, a random number between three and five annotations from randomly picked users which already annotated the 3D-model and land cover map used in the current game round are chosen and stored for further processing. As the pool of 3D-models and land cover maps to choose from is big, the possiblity that a player already played the same combination of model and maps earlier is relatively low. Thus, the set of data retrieved from the database for each game round and assigned to an emulated player may come from different users. Having this set of data for each emulated player, it is now possible to pursue the game procedure as if it were a regular multi-player game. Even player names are stored for emulated players to show them at the end of the game. As a matter of course, the annotations stored for emulated players are matched with those of the human player and are available for rating, but are not stored to the database again. In a regular game, the game countdown speeds up if a player typed in the maximum number of annotations possible and the game round ends if all players are finished. This game element is emulated too by retrieving and adding up the users average time taken to enter each annotation from the database, while three seconds are added for each word missing to the maximum number of annotations possible. If all emulated players and the human player is finished, the game round ends and the scoring round is loaded.

Pioneer Mode - Strict Single Player

But this asynchronous game mode only works, if the chosen models and maps have already been used in other game-rounds before. If new models or land cover maps are added to the database, no single player game would be possible. That is because the lack of user data would inevitably lead to zero points, as the annotations entered by the player would not match any others when there are no words to compare to. For that reason, a so called "pioneer mode" has been implemented. This mode is a strict single player mode, letting the player know that no other players participate in the current game. The player is informed that he or she is playing alone and that the data will be stored for further games. So each game-round is played like in the regular game mode, but the player does not have to wait for others to finish. If the maximum number of annotations were entered or 100% of the canvas has been painted, the game round is finished and the game switches to the score screen. If the current player is the first to play a particular pioneer game, a temporary score is shown for each annotation and drawing. At the end of the game, a message is shown to the user that the input has been saved and that it will be used for further pioneer players. So whenever a pioneer game starts, it is checked if there are pioneer games available which were played only by one user yet. If available, the user is joined to that game and plays just like the first player. At the end of the game, all the input from player 1 is fetched and compared to the input of the current player to calculate agreements and thus scores. The final score is then shown to the second player, just like in a regular game round. The difference of the temporary and the final score is calculated and added to the database. The next time the first player joins the game, he or she is notified about the gain or loss of points due to a finished pioneer game. The pioneer game is triggered randomly and only if only one user joined a game.

3.1.8 Anti-Cheating-Mechanism

All the measures described above do not protect the game from cheating mechanism or players which intentionally enter wrong annotations or draw wrong classificiations. For example, users could agree on entering "aaa" all the time. This way, user input would match and all players involved would get points for it. Likewise if players agree to scribble the canvas with the same category all the time, the scoring would be influenced. Single players drawing wrong maps or entering nonsense annotations intentionally to try to game the system are a problem too. The cheating strategy explained however is unlikely to be effective, because players are randomly paired together and have no information about who they are playing with. The possibility that two players, who agreed on a cheating strategy before, are paired together should be low. Additionally, communication channels like chats have been omitted intentionally, because having the possibility to chat, agreeing on cheating-strategies would be trivial.

3.2 Differences to Related Work

Here, the differences in terms of game mechanics between TAGinator and related work as described in Chapter 2 is treated. It has been already stated that the game consists of three game-

rounds. These aim at collecting annotations for 3D-models and land cover maps as well as at collecting data about the different types of land cover by letting users draw on maps.

Like in the ESP game described in Subsection 2.2.1, players have to enter annotations which agree with those of other players to get scores. These scores are shown on the start page in form of highscores to motivate player to compete with others and to keep playing the game. But the concept is different in many ways. First, we do not collect annotations for images but for 3D-models and land cover maps instead. The advantage of this is that we can collect different data within the game and are not limited to images while we give variety to users. Another difference lies in the approach of validating annotations. In the ESP game both users enter annotations until they match. Then, another image is shown until the time is up or the players have agreed on 15 images. In our game up to five users can play at the same time and each player can provide up to five distinct annotations per model or map until the countdown expires. The annotations are then compared to find matches among the users. This way we can collect more annotations in general as we do not abort the annotation process once a match is found. Additionally, the more players agree on the same annotation the better it tends to be.

While the ESP game is limited to collecting annotations, our game additionally aims at letting users draw on maps to categorize land cover maps. Furthermore it fulfills the aforementioned high-level heuristics for social games to give incentives to players to recur to the game. This includes social interactions like posting messages to the Facebook wall or invite friends to the game. Generally speaking, the difference lies in the platform. The ESP game is hosted on the website www.gwap.com while our game is integrated into Facebook to circumvent the need to register or login to an external page to play the game. Additionally, users are already used to the Facebook look-and-feel. In contrast to the ESP game players are able to make progress within our game by providing achievements that can be collected. Users of the ESP game can however invite friends to their platform to get scores, but further achievements are missing.

A part of the ESP game's mechanics are taboo words. Taboo words are annotations which have been entered by many users before and are therefore blocked for further games. This forces players to provide different annotations but the most common ones. The more taboo words are shown for an image, the more points the players get for agreeing on an annotation. This approach is not implemented in our game as the images shown in the ESP game mostly show several objects or motives and thus have various possible and correct annotations. 3D-models are in contrast more specific and the number of potential annotations is thus more limited. The maps used in our game show mainly more or less unique types of land cover. For example, if a map shows an area covered with shrubs, it is not reasonable to block the word "shrubs" for subsequent games. We also expect that the chance to agree on annotations would significantly decrease if taboo words would be provided.

There exists no official information about a single player mode where a missing player would be emulated, but as bots instead of players can be inserted to prevent users from cheating, it is safe to assume that the ESP game can be played even if no other player is online at a given time. Another similarity is that both games are implemented using JavaScript to enable a platform- and browser-independent game experience. The ESP game as well as our game works nicely with modern mobile devices. Also sound effects are implemented in both games and ingame chats have been omitted.

3.3 User Interface

The game starts with a page where highscores and achievements are shown as well as links to a tutorial page and to the game itself. The starting page is shown in Figure 3.4. Whenever a player

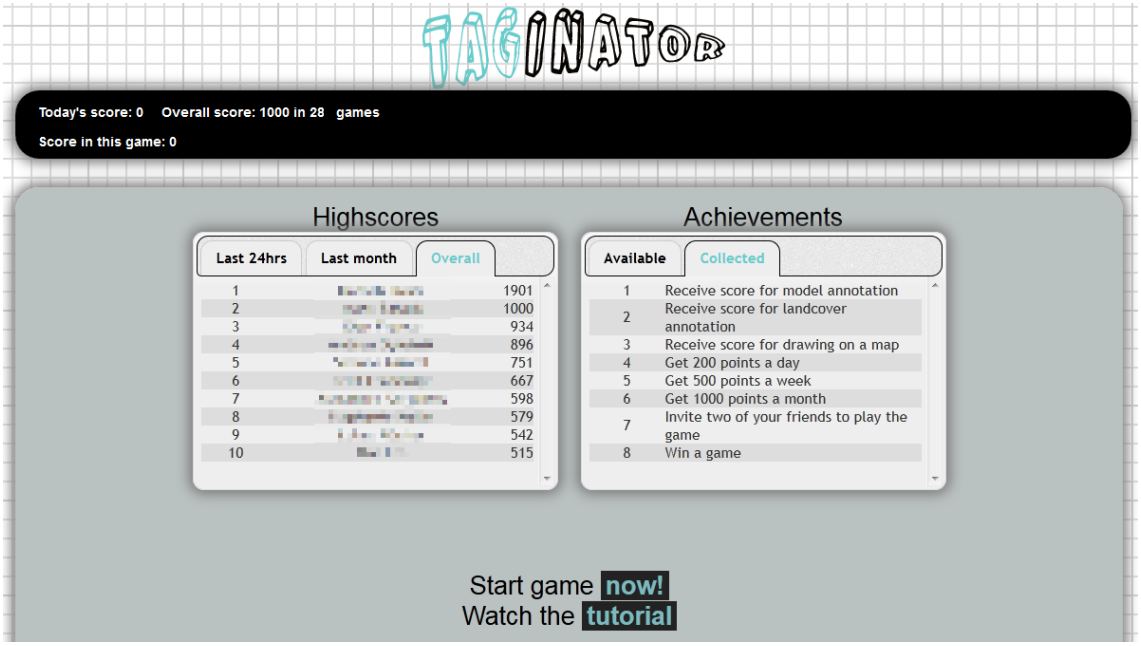


Figure 3.4: Starting page of TAGinator. Here, highscores, achievements and links to the game and a tutorial is shown.

joins the game, he or she is connected to an open, not yet started game, called a room. If no room is found to join to, a new one is created and a countdown starts. Other, waiting users are then paired until the countdown expires. The game starts immediately after the countdown has expired, the room is closed for further joins and an event is broadcast to all players connected to that room to load the game screen. The game consists of three independent game-rounds, that are played in a randomized order: one for annotating 3D-models, one for annotating land cover maps and one for drawing on a map to categorize land cover. First, the general layout is covered in this section and all game screens afterwards.

3.3.1 General Layout

It has been mentioned before that this thesis lays its focus on channeling human brain power to solve useful tasks. This can be achieved by combining the ideas of crowdsourcing and social games to reach as many users as possible. Consequently, a game is needed which offers a simple but entertaining gameplay to enable playing in short, sporadic bursts without barriers while emphasizing socializing at the same time. This includes accounting for different screen sizes and resolutions as well as operating systems and browsers. To stay abreast of changes and the steady growth in the market of mobile devices, we put emphasis on developing a game design

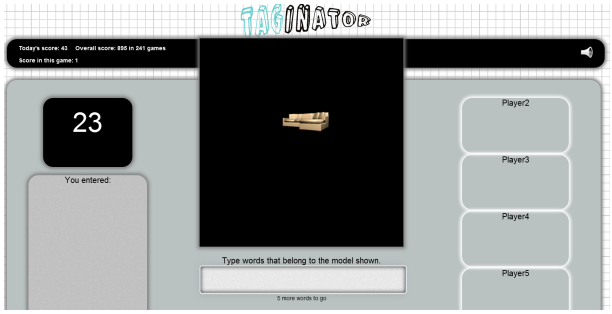
which is useable in the same way for every user. Hence, the layout is done using Cascading Style Sheets (CSS), because it adapts to different screen sizes and can be adjusted to fit even smaller screens. This is especially important for today's mobile phones, because almost every modern smartphone supports changing between landscape and portrait orientation by tilting the device while screen size is limited. By using only relatively aligned containers without fixed dimensions, the layout can always accomodate itself to given and changing screen resolutions or aspect ratios. Figure 3.5 shows screenshots of the game on a 24“ desktop monitor in full screen and for half screen width as well as for an Adroid smartphone, an Apple iPhone and an Apple iPad in landscape orientation. It can be seen that the layout looks very similar, even if screen size and resolution are subject to considerable fluctuations. The portrait orientation is not shown here, as it looks identical compared to the layout in full screen on the monitor.

The layout in general has been intentionally held simple to reduce problems with different resolutions and has been designed to fit the limited space given on mobile devices and for integration into the Facebook platform. Mostly, a straight linear layout is used to neatly arrange game elements. Elements include the logo, horizontally centered on the top, the header below as well as the main container in the middle of the page and the footer on the bottom, each justified and stretched to cover 80% of the width available. User scores are shown in the header, including the score achieved in the latest game, today's score and the overall score of all games played.

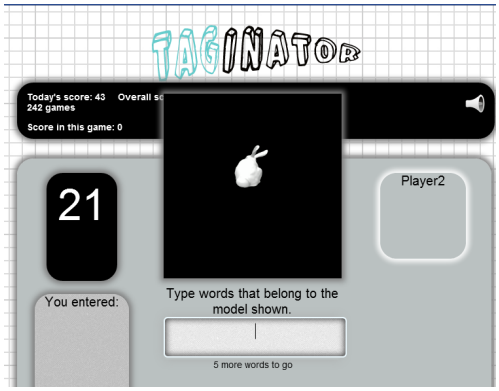
The look-and-feel is sportive, supported by a checkered background, which is reminiscent of a sketch block. Additionally, the vast majority of shapes used have rounded borders and the logo uses a comic-style font to complete the impression of a sketch. The colors are used in a way to maximise contrast and quiet colors are used for borders and backgrounds of dynamically generated elements within the game flow. Colors are chosen because of their psychological and physical effects. While red has been shown to raise blood pressure, but is not beneficial for work and calls attention, green and blue-green colors have a relaxing effect and have shown to promote work where concentration is required [23]. Thus, elements like speeded countdowns or players disconnecting are illustrated in red, while the regular game flow is accompanied by grey-scale colors and cyan. Figure 3.6 shows the basic layout on the example of the idle screen. The game consists of three independent game-rounds, where each round is designed to collect different data. Those game-rounds and their user interface elements are explained in detail in Section 3.3.

3.3.2 Model Annotation

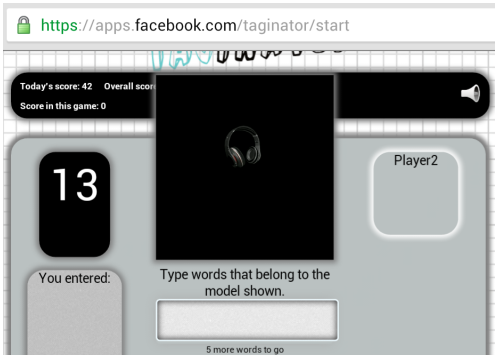
The game-round for annotating 3D-models is composed of three horizontally aligned containers. Figure 3.7 shows the round for model annotations and illustrates the elements of the user-interface which are described and numbered int the following. The very middle of the screen shows a rectangular area, where the video of a randomly chosen 3D-model is shown in an endless loop, rotating around its up-axis (4, see Figure 3.7). Showing a video has many advantages over loading and rendering the model directly into a HTML5 canvas element using WebGL. These advantages are the considerably smaller file size and thus loading time as well as support for mobile devices, as the large majority of these do not support WebGL yet. More details about this can be found in Section 4.1. The video is loaded in WebM format, with MP4 as a fallback if



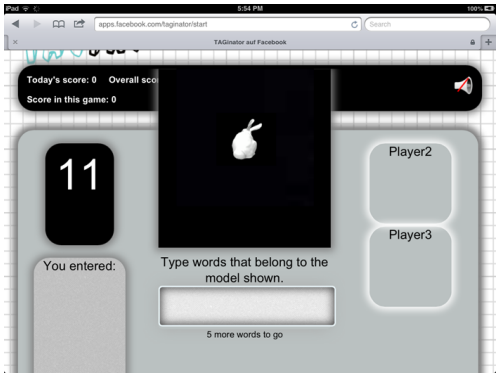
(a) 24“ widescreen monitor in fullscreen



(b) 24“ widescreen monitor with browser window resized to half the screen width



(c) Samsung Galaxy S3 in landscape orientation running Chrome on Android



(d) Apple iPhone 3GS in landscape orientation running Safari

(e) Apple iPad running Safari

Figure 3.5: This figure shows the same game layout on different resolutions, screen sizes and platforms.

WebM is not supported by the browser. Right below the video container, there is a textarea (1), where players can enter and submit annotations for the model shown. Each player can provide

Figure 3.6: Basic layout of TAGinator. This screen is visible at the beginning while waiting for players to join.

up to five distinct annotations which are then stored, as long as their character count exceeds three and the annotation has not been entered by the player in the same round.

When a player has provided the maximum number of annotations allowed, the textarea is locked to prevent the player from entering too many annotations. On the left side of the screen, there is a countdown which shows the time left for this round to provide annotations (5). Whenever the first player entered the maximum number of annotations possible, the countdown fastens and turns red to hurry up the game round. Right below, there is a box where all valid annotations of the player in the current game round are shown (2). On the right side of the page, fellow players are listed (6) so that the player knows how many people he or she is playing with. Whenever a player finishes, an overlay in the form of a stamp is added (7) to that specific player's container for every fellow player to mark which players are finished and which are not.

3.3.3 Land Cover Annotation

The second game-round is designed to collect annotations for land cover maps entered by players. The layout is basically the same as for the 3D-model annotation round, except that there is no video shown in the middle of the screen. Instead, a Google Map is shown, which is centered on given coordinates in Latitude and Longitude. The map is shown in the maximum available and useable zoom-level and is overlaid with a red rectangle, covering an area of $4km^2$. Zooming in and out using the mouse wheel is enabled, as well as dragging the map with the mouse. As for

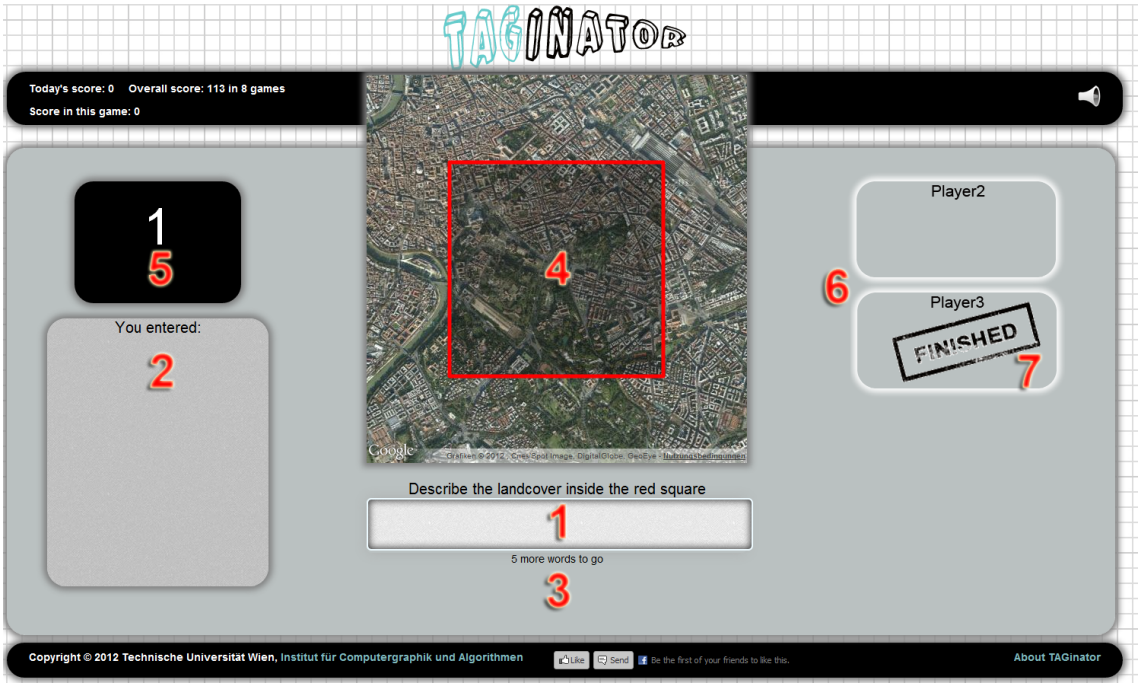


Figure 3.7: Game-round for the annotation of 3D-models with numbered elements

the 3D-model annotation round, all players can enter annotations, until the countdown expires or the maximum number of annotations possible has been entered.

3.3.4 Land Cover Categorization

The third game-round enables the player to categorize land cover maps by a drawing. Figure 3.8 shows this round with every element that has a number in the folowing description is numbered. To enable user-drawings, a Google Map as in round two is loaded (1, see Figure 3.8), overlaid with the same rectangle as in the round before and zoomed to the maximal available and useable zoom-level. User interface controls like zooming with the mousewheel or dragging the map are disabled. The element containing the map is then overlaid with a HTML5 canvas element, where the player can draw into. The canvas has exactly the same size as the area inside the red square, which is achieved by a conversion from Latitude/Longitude coordinates to pixel values, to calculate the position of the canvas. For that reason, the players are not able to change the map by zooming or dragging, as this would require to re-initialize the canvas, which would result in losing all data. To draw into the canvas, the player simply has to left click into the canvas or drag the mouse over it. At the cursor position, a circular snippet in the size of the cursor of a pre-loaded template corresponding to the selected category is inserted (2). This enables uncovering a smooth category-image on a specific location instead of having multiple copies of the same icon.

Categories can be selected on the left side of the page (3). Each category is presented with

its name and a small icon, to help the player chose the right category he or she is searching for. Because areas of distinct categories vary in size or shape, different brush sizes are available on the right (4) to support either large-scale or detailed drawings. Every time the player releases the mouse button, the canvas is analyzed and the percentage of pixels filled is calculated. On the top right of the map, a circular element shows this percentage (5) to indicate how much area still needs to be filled to gather a reasonable result. The color of the number changes from red to green, when the threshold is exceeded. For percentages below the threshold, the data is discarded at the end of the game round, as we do not expect to get reasonable results from such data. As in the game-rounds before, a countdown (6) indicates the time left in this game round to finish the drawing. As we do not want to overly hurry players while drawing, the countdown does not speed when another player has finished his or her drawing. Additionally, fellow players are not shown in this round to free space needed to show categories in a neat way.

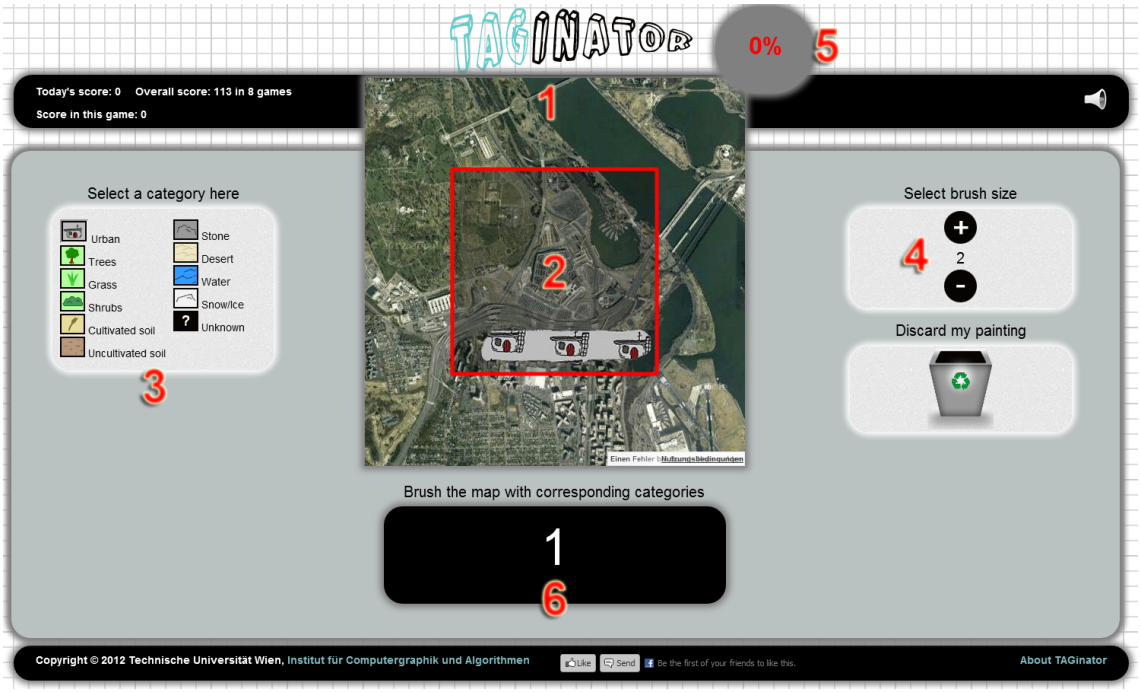


Figure 3.8: Game-round for categorization of land cover maps by user drawings

3.3.5 Score Screen and User Rating

Whenever a game- or scoring round is finished, an animation is started to playfully change the content from game round to scoring round or vice versa. This animation slides the currently middle container out of the screen and subsequently a new container is slid in from outside the screen on the other side. This is achieved using a proper design which allows elements to float. Using CSS properties, the positions of the sliding elements are gradually changed until the initial positions have swapped. The element sliding out of the screen is removed, when it has left the

screen completely. Figure 3.9 shows an interim stage of the animation from game screen to score screen.

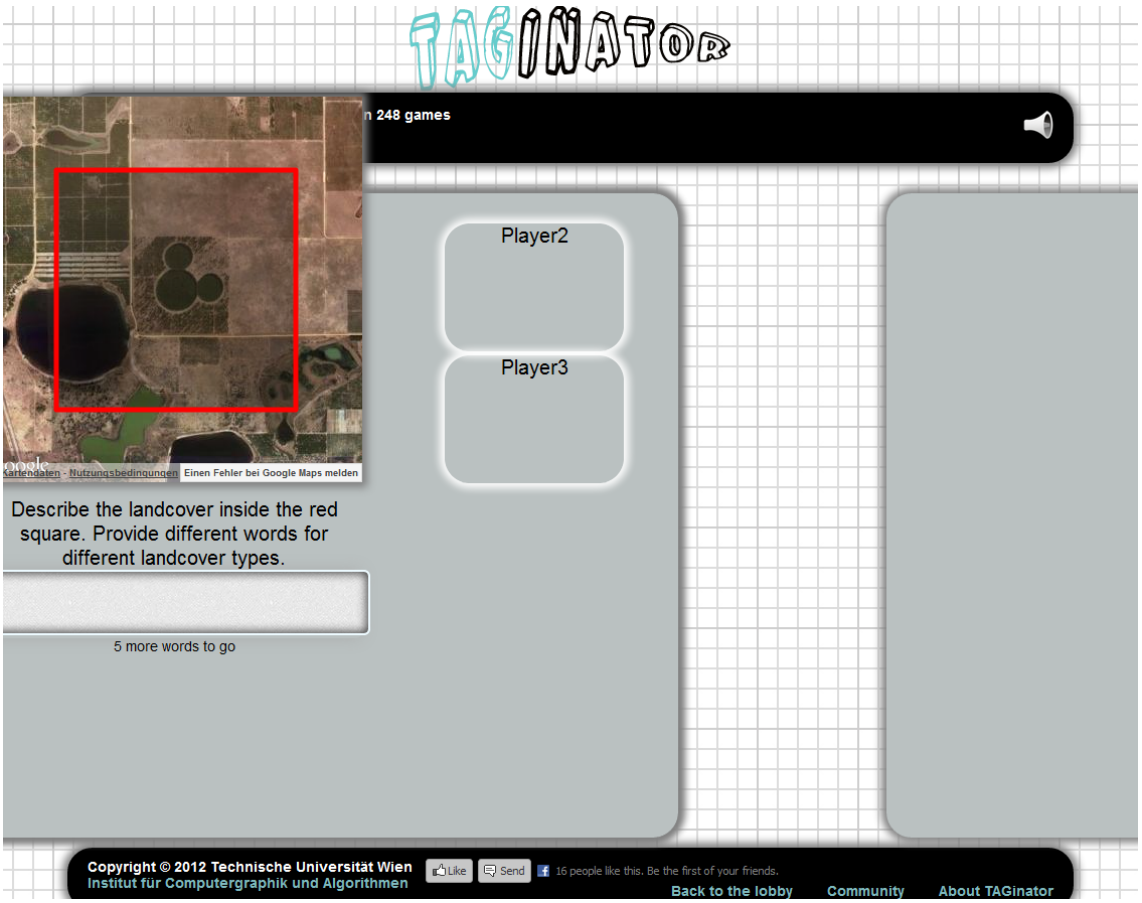


Figure 3.9: Sliding animation between game screen and score screen

After the animation has completed, the score screen is loaded. Each player now sees all the annotations he or she entered in the prior game round, together with points and bonus points achieved (1). Additionally, scores of fellow players are shown (2), but without the corresponding annotations. In the right table, all distinct annotations from other players are shown. Each annotation has two buttons, which enable the user to up- or downrate annotations adjudged to be good or bad (3). To avoid uprating ones own annotations, only those of fellow players are shown which have not been entered by the user. If the user rates an annotation, the rating is matched with the number of good and bad ratings for that specific annotation stored in the database. Depending on the algebraic sign of the difference of good and bad ratings, it is decided if the rating corresponds to the ratings stored. For every rating which matches the stored rating, one point is given to the user, which is indicated by adding a small icon on the cursor position for a short time. Because we wanted to avoid tactical rating, players do not get penalized if their annotations get rated bad, but are also not rewarded for good rated ones. Each player can up-

and downrate two annotations each. At the bottom of the screen, there is a countdown which shows the time left in the score round. For the drawing round for land cover categorization, there is no possiblity to rate other players drawings because of the lack of qualitative features which could be judged by players. Thus, only the scores are shown and the countdown takes less time to expire. Figure 3.10 shows the score screen while rating is enabled. Every element described above is numbered.

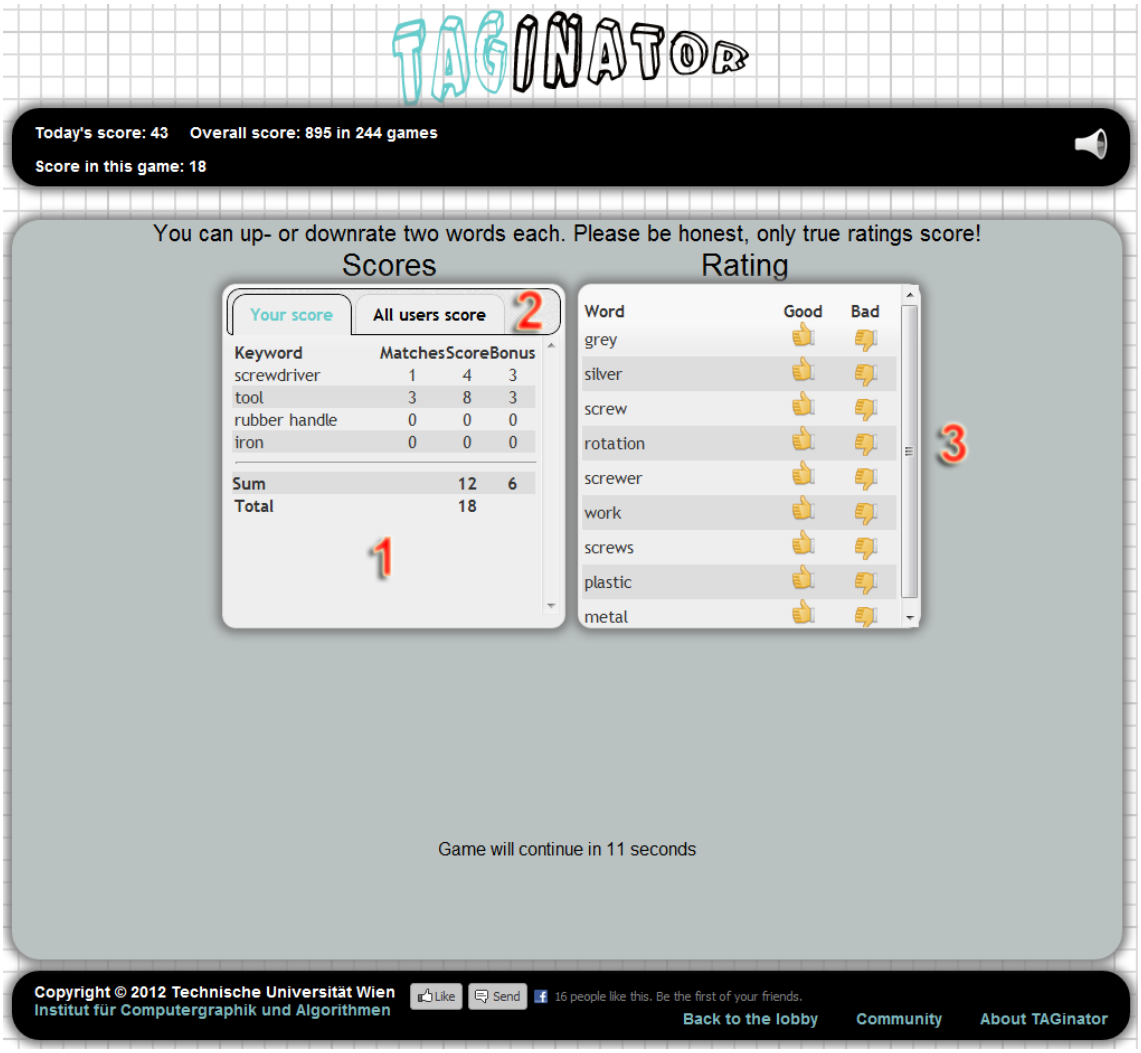


Figure 3.10: Scoring screen with buttons to up- or downrate good or bad annotations

Once all three game rounds were played and the score rounds are finished, the final score screen is loaded. Here, for each player a container is arranged in an elliptical layout. The player's own container is highlighted with a background image, so that it is clear which container shows ones own score. Then, an animation is started which changes the opacity of the containers one after the other a couple of times. During the animation, scores for each player are incremented

until the final score is reached. Then, for the first time in the game, the names as well as the Facebook profile pictures of all fellow players are shown to amplify team spirit and the social aspects of the game. The player, respectively the players with the highest scores, are highlighted to indicate the winners of the game and the other containers fade from the spotlight. After that, buttons are shown which provide the functionality to post a message to the Facebook wall, showing that the user achieved his or her particular score while playing the game. It is also possible to invite friends to play the game, which will be honored by an achievement. The choice to post on the Facebook wall or send invitations to friends is indeed up to the user. There is no possibility to add fellow players as friends, as this is not allowed by the Facebook rules. Figure 3.11 shows the final score screen.

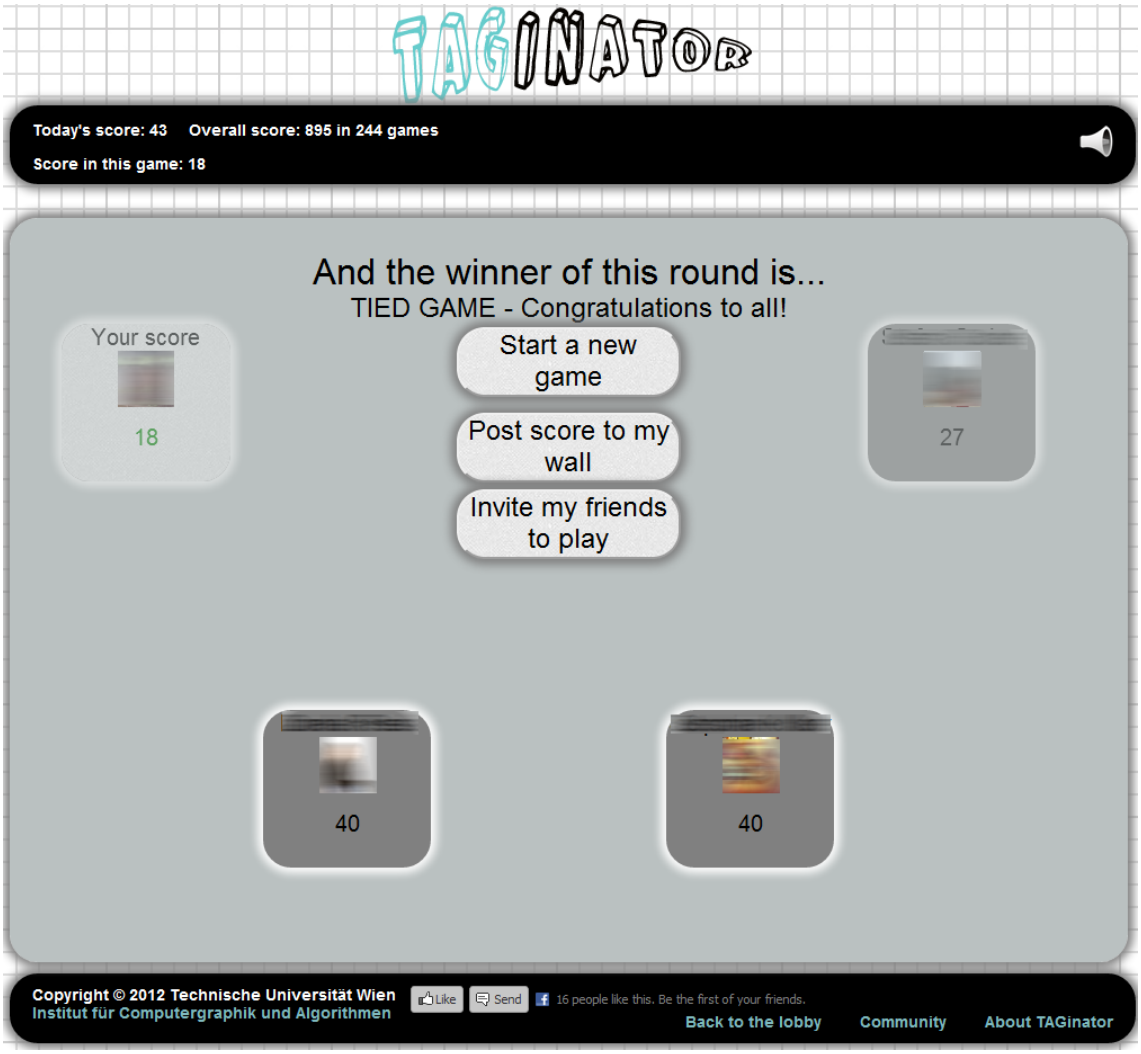


Figure 3.11: Final score screen with highlighted winner

CHAPTER 4

Technical Background

This chapter covers the problems regarding the game mechanics that occured during the implementation and how they were solved.

4.1 Support of 3D Models

We decided to implement support for 3D models, as HTML5 combined with WebGL is capable of rendering 3D graphics. The motivation for this decision is given in detail in Section 1.1. The implementation started with loading 3D models from the Collada format and rendering them directly to a canvas element using WebGL. Despite the fact that this approach worked out for the latest desktop versions of Mozilla Firefox and Google Chrome, it has two drawbacks. First, the Collada file needs to be transfered to every client participating in the game. This is problematic, as the file sizes are subject to wild fluctuations. Even if the bandwidth provided by internet service providers is increasing continuously, one can not rely that every user has a high bandwidth available. The other drawback is that the vast majority of mobile browser do not support WebGL yet. Even the latest version 18 of Google Chrome mobile browser on Android 4.0.4 does not support the 3D context needed to render 3D graphics using WebGL.

For that reason, we discarded the approach of rendering models directly into the canvas. Instead, we are now rendering the models into video files as a preprocessing step. For each model, videos in both MP4 and WebM format are created. The videos are five seconds long with 25 frames per second and cover one complete rotation around the up-axis to enable seamless looping. By following this path, the video files are nearly constant in size and significantly smaller than most of the model files, which saves bandwith and loading time. The resulting videos in WebM format are about 33% smaller than in MP4 format, but as a fallback for the case that WebM is not supported, the video in MP4 is used. Another advantage in rendering the model to videos instead of rendering them directly to the browser is that the video can be shown on mobile devices whose browser support the HTML5 video tag and the WebM or MP4 format. This is the case for the latest versions of the mobile versions of Chrome, Firefox and Opera as

well as the default Android browser and Dolphin Browser HD, which is available at the Google Play Store. It has also been tested and verified on iPhone and iPad using Safari.

With the approach stated, it is now possible to present videos of 3D-models which can be textually annotated within the game. The gameplay itself is described in detail in Section 3.1.

4.2 Support of Land Cover Maps

Besides showing 3D models, the game also supports the textual annotation of land cover maps as well as the categorization by drawing on a map. These maps are displayed by making use of the Google Maps API v3, which is designed to manipulate and embed Google Maps in websites and to add content to these maps. It is free to use, as long as the resulting website is also free to use and publicly accessible.

4.2.1 Creating the Maps

We first need to create a map which is centered at coordinates given in Latitude and Longitude. These coordinates are provided by the Geo-Wiki project and define regions that show large differences among different land cover maps. Now a zoom-level is selected that delivers the best resolution possible while fitting the game layout. This is done asynchronously with the Google Maps API which returns the highest zoom-level available or times out if the zoom-level cannot be determined. In case of a timeout, a fixed level is used as a fallback. To uniquely identify the region of interest, a square is added to the map that covers an area of 4km^2. It should be noted that it is not possible to add a square overlay with a given side length specified in km. This is because the API does not allow to specify the rectangle's bounds in meters, but only in Latitude/Longitude coordinates. Therefore, a circle with a radius of 1km is added and its bounds are used to create a quadratic overlay. This can be done because the API allows to specify a radius in meters for circular overlays. This is illustrated in Figure 4.1.

The code to do that is simple, but may be of interest as it is not documented so far.

```
var map = new google.maps.Map(
    document.getElementById("containing_div"),
    someOptions
);
var circle = new google.maps.Circle({
    center: someOptions.center,
    radius: 1000
});
var square = new google.maps.Rectangle({
    strokeColor: 'red',
    strokeWeight: rectangleBorderWidth,
    bounds: circle.getBounds(),
    map: map
});
```

Figure 4.1: Land cover map with low resolution. The bounds of the circle are used to create a quadratic overlay.

As we are interested in land cover data, the map type is set to “satellite” instead of map or hybrid mode. The default user interface is disabled, as the control elements would distract from the essential task and do not fit the layout, but zooming via the mouse wheel is also enabled as dragging the map with the mouse. The map is now used to present land cover maps to users so that they can textually annotate the land cover shown.

4.2.2 Drawing on a Google Map

As for land cover annotation described before, a Google Map is loaded with given coordinates, zoom-level, center and bounds and a red rectangle is added to mark the region of interest. To enable user drawings, the map is overlayed with a transparent HTML5 canvas, which exactly fits the size of the quadratic bounding overlay added to the map. The HTML5 canvas object is basically a rectangular container, which can be used to draw graphics using JavaScript. It provides a variety of functions to draw primitives, text or paths and to copy images or parts of images into it.

4.2.3 Conversion between Map Coordinates and Screen Coordinates

Because the area in pixels which is covered by the red rectangle depends on the zoom-level of the map, the dimensions of the canvas overlaying the rectangle needs to be calculated. The rectangle’s corner points can be accessed by the API, as it is an overlay of the map. However, these points are given in Latitude/Longitude coordinates. That means that we have to convert these coordinates into pixel values to create a canvas which fits the area of interest marked by the rectangle. Unfortunately, the API version 3 does not provide a straight-forward way to do this, as it had in version 2. Instead, a canvas projection overlay needs to be created, which is basically a sub object of the Google Maps OverlayView. With this canvas projection overlay, it is now

possible using the API to convert the quadratic bounds given in Latitude/Longitude coordinates into pixel values relative to the container the map is rendered into.

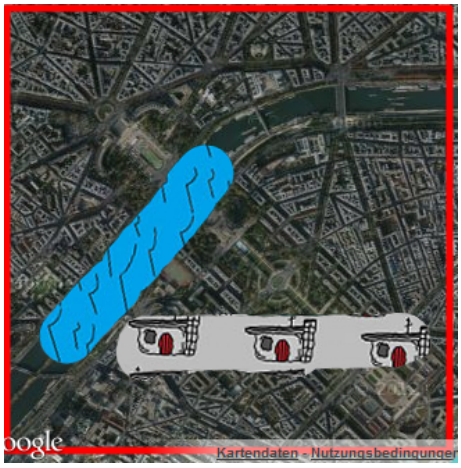
4.2.4 Drawing into an HTML5 Canvas

Whenever the user clicks on the canvas, the center pixel and the pixels in a surrounding area depending on the brush size chosen should be marked as belonging to the category chosen by the user. To do this, first the width and height of the affected area needs to be calculated. Then, the icon corresponding to the category chosen by the user is clipped to get a circular brush-shaped image which is then copied to the right position. But having only one canvas element containing circular parts of category icons does not allow to store which category was chosen, so a second canvas and 2D context needs to be created. This context is then filled with a circular area too, but only the red channel is filled with unique values for each category to accelerate further processing. This is shown in Figure 4.2c. For the data canvas and context, it is not sufficient to use the HTML5 functions to copy circular regions, because these functions also include anti-aliasing, which leads to wrong color values near the circle border. As this feature cannot be deactivated, we need to check manually for each pixel if it lies inside the brush-circle.

If icons are just copied centered to the cursor position, the result is a strong repetitive pattern shown in Figure 4.2a. Furthermore if the user drags the mouse or clicks on neighboring pixels, the icon will overlap and the resulting image is not recognizable anymore. Instead, templates for each category in the size of the canvas are loaded and parts of them are copied to the 2D context. These templates consist of smooth repetitions of the category-items. This gives a seamless drawing even if neighboring pixels are selected as shown in Figure 4.2b.

As mobile browsers support gestures like pinch and zoom, dragging, or zoom on double tap, ordinary mouse event listeners are insufficient. For that reason, the Javascript API supports event listeners specifically designed for mobile devices called “touchend”, “touchstart” and “touchmove”, which differ in functionality from their mouse counterparts. By using them in addition to mouse event listeners, the default behaviour of mobile browsers for user gestures can however be circumvented. It needs to be said that the default user interface of Google Maps including zoom, drag and scroll is fully disabled for the drawing-round, because changing the area of the map shown would make it necessary to completely reset both canvas and 2D contexts to ensure that both map and context match exactly, which would lead to the loss of data specified by the user so far.

(a) This is the result of copying an icon to the cursor position when the user drags the mouse or clicks neighboring points.



(b) The correct way is shown here. A part of a pre-processed template image is copied instead of repeating an icon.

(c) Each category is assigned a unique value for the red channel. For demonstration purposes, the content of the canvas has been overlaid with the map.

Figure 4.2: This figure shows the data of canvas 2D context for the wrong (a) and correct (b) way of filling in user paintings as well as an example for the unique values for each category for the red channel (c).

4.3 Client/Server Synchronization

Having different game modes to collect different data requires multiple game-rounds. It is therefore required for a multiplayer game to synchronize the game flow so that every user solves the same tasks at the same time. This is essential, as the game is based on agreements on given annotations and drawings with alternating game and score rounds. This will be explained in more detail in Section 3.1. To achieve a synchronized game experience, the server triggers the loading of game and score screens. For that, the server broadcasts messages to each client connected in a game. All clients receive these messages and manipulate the Document Object Model (DOM) to show the correct game screen. This way, page refreshes become unnecessary, which makes an interactive game experience possible.

Whenever a client finished loading a screen, it sends a notification to the server. The server collects these notifications and triggers the next action when all notifications have been received. This works well as long as connections are not interrupted. If a client disconnects due to connection problems or just a page refresh, or if the client closes the window, no notification is sent and the server would wait forever to trigger the next action. The easiest way to handle this is to remove the client from the userlist whenever it disconnects. But the asynchronicity of the communication between server and client is a challenge for these problems. If the last user whose notification is still missing disconnects while all other clients already sent their notification, a deadlock happens. The client indeed gets disconnected and thus all clients sent notifications, but the server still waits for the last notification because at the time the notification of the second last user has been sent, not all clients were finished.

To handle this problem, callback queues have been implemented. Whenever the first client sends his or her notification, a routine is started which checks in a given interval if all clients sent their notification, unless all notifications have been received. If the routine is called a given number of times, all clients whose notification is still missing are disconnected to prevent other users from waiting too long. Each client sends a callback function to the server which is stored in a queue and called when all clients finished, with or without disconnecting others. That way, even client side problems can be handled where both notifications and disconnection events never arrive.

Another problem which crashes the game if not handled correctly are multiple connections from the same user. This is a problem, because the socket is registered with the user session. If the same user connects multiple times, multiple players may be joined to multiple different games, while the socket remains the same. This would result in cross-communicating sockets, sending messages to all games the user has joined in. Because of this, the server could trigger actions too early. Imagine the socket from game 1 sends the notification that the client finished the game round while the round is not yet finished in game 2. To handle this, a global list of active players is stored and everytime a user connects, it is checked if the user is already connected. If yes, further registration of the player is stopped and a message is shown to the user.

4.4 Portability

Following the approach of exploiting the advantages of social platforms has its drawbacks: the game needs to run in a web browser. Hence, the technologies available for the implementation are limited. A decision between two technologies, namely Adobe Flash™ and HTML5 had to be made. Adobe Flash™ has been an inherent part of the internet in general and also in web development for the last 15 years. Advantages of Flash are the possibilities in user interface design as well as the fact that interactive websites are possible without the need of page refreshes. But the disadvantages outweigh the advantages for our needs. Flash movies cannot be optimally indexed by search engines and other features like changing the font size are missing, as the font is embedded into the Flash movie. But the most important fact for this thesis is that there is no support for Flash in Apple's mobile operation system iOS™ which is running on iPhone and iPad. Google's mobile platform Android supported Flash since the beginning, but Adobe has announced that there will be no further support for mobile devices from Android 4.1 onwards. In August 2012, Android and iPhone together have a market share for mobile devices of about 85%[18]. Using Flash, all of these mobile devices would be locked out from our project, which is not in harmony with the approach of reaching as many users as possible.

So we came to the decision to use HTML5 for our project. The definition of HTML5 is a bit fuzzy, as it describes both the specification of the fifth version of HTML, but also a set of technologies [4]. Over 60 different APIs are included currently and the number of incoming requests to add more still rise. However, the most important innovations of HTML5 for us are the canvas- and the video-tag. With the HTMLCanvasElement it is now possible to render directly to the screen without using other technologies or frameworks. There is a vast number of possibilities to use the canvas element. Basically, there is a built-in object for 2D context which offers a variety of functions for drawing 2D primitives as well as text and images. Since 2011, almost all major browsers – except Microsoft Internet Explorer – support WebGL, a graphics library for web browsers. With WebGL, it is possible to render hardware-accelerated 3D graphics directly to the browser's canvas. It is based on OpenGL ES 2.0 and is implemented in the rendering engines Webkit and Gecko, which in turn is used by browsers like Firefox, Safari, Chrome, Opera and others. The video-tag enables the browser to play videos directly, without the need to rely on plugins of video players. Until HTML5, there was no standard for showing videos on a web page. Another big progress is that HTML5 is much more developer-friendly than earlier versions. The specifications are much tighter in order to eliminate the prevalent need to adapt an application to a variety of different browsers, each interpreting tags or stylesheets differently [4]. Because browsers do not stick to standards, the results are broken layouts or even missing functionality. That is in general a big problem, because the need to fix layout- and functionality problems for every browser still in use is a very time-consuming and thus costly task. However, HTML5 is not yet a standard, but it is implemented in all latest versions of major browsers and signs are pointing to HTML5 becoming a standard soon.

CHAPTER 5

Results

This chapter covers the results and insights gained by analyzing the collected data. To get these data, a small user study has been performed, asking users to play the game as long as they want. To collect meaningful data, the set of models was reduced to 19 and the number of land cover maps was reduced to 20. A total of 36 users played the game, each player played ten games on average. As already described in Subsection 3.3, the game consists of three game-rounds, each round collecting different data.

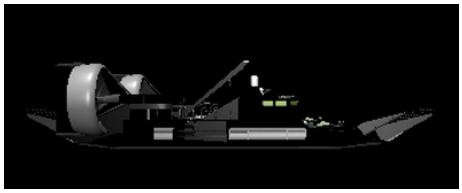
Round one aims at collecting annotations for 3D-models. A total of 1,376 model annotations were collected, 400 of it are distinct. Round two shows a land cover map instead of a 3D-model and users have to enter annotations as well. 247 distinct annotations out of a total of 1,280 were collected during the user study. Round one and two allows the user to enter up to five distinct annotations. In the third round, users are asked to portray a land cover map by choosing categories that correspond to the land cover shown and paint on the map, which led to a total of 294 drawings.

The results of the user study are analyzed in more detail in this section and its subsections. All results in terms of diagrams can be found in Appendices A, B and C. All land cover maps are generated using the Google Maps Javascript API V3. Thus, the copyright holder of the maps is Google inc. It should be noted that we show results for all annotations entered by the users as well as for those annotations entered first. This allows to analyze the importance of descriptive annotations like color. For the analysis of the annotations entered first we use only about 70% of the data collected within the user study. This is because we needed to filter the data to get the right order of user inputs to select the first one. However, those first-mentioned annotations can be compared to the total set of annotations because the differences in the results due to the different set of data are negletable.

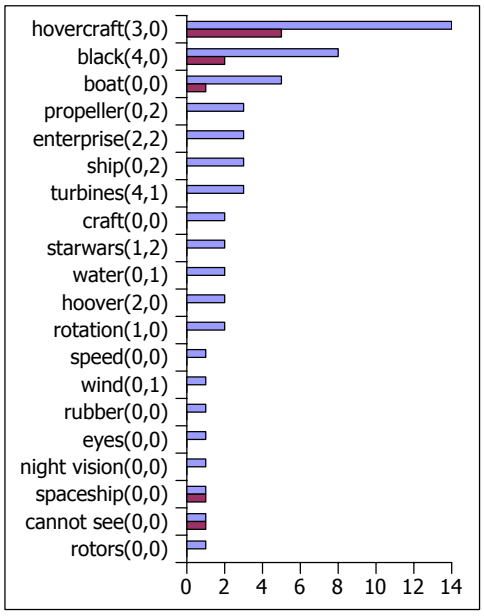
5.1 Analysis of Model Annotations

In this section, we are going to analyze the data collected in the first round of the game. We intentionally chose models in different qualities to examine if the quality of the results depends

(a) 3D-model of a hovercraft with bad contrast and quality



(b) 3D-model of a hovercraft with enhanced contrast for illustration



(c) Annotations and user ratings for 3D-model of a hovercraft with absolute number of occurrences in blue and number of occurences for first entered annotations in red

Figure 5.1: This figure shows a 3D-model of a hovercraft (a) with its top 20 annotations and user ratings (b).

on the quality of the models itself. As the analysis of all 20 3D-models would be beyond the scope of this thesis, we exemplarily show the results of three models with different qualities in this section and explain the insights gained from the results.

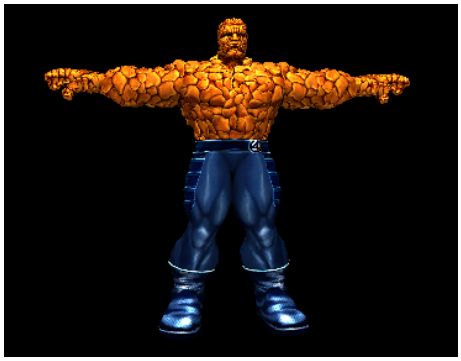
5.1.1 Low-quality Model

The first example is the model of a hovercraft that has both low contrast and low quality. It has been intentionally chosen to answer the question if and how much the quality of annotations depend on the quality of the model itself. The contrast of the model in Figure 5.1a which has been used in the user study is enhanced, so that it can be shown in this thesis. It can be seen in Figure 5.1b. From a total of 68 collected user annotations, 20 out of 31 distinct keywords are shown in Figure 5.1c. It plots annotations and the corresponding user ratings on the y-axis and the absolute number of occurrences on the x-axis, where the maximum of the y-axis is set to the count of the most common annotation. The blue bars indicate the distribution of all entered annotations while the red bars show the distribution of only the first annotation entered by users.

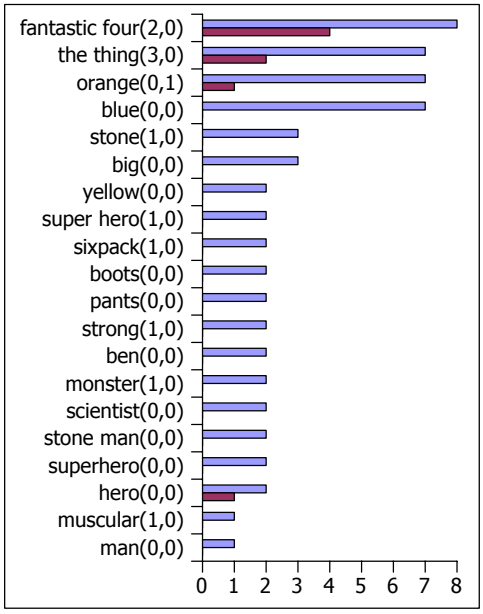
One can see that the top three annotations cover about 40% and the top five about 49% of all entered annotations for this model. Even if quality and contrast are low, the top annotations doubtlessly match the model well. The ratings in brackets were collected in the scoring round after the annotation round. Each user can up- or downrate other players annotations, but it is not

possible to rate annotations which were entered by the user himself in the same round. To prevent excessive up- or downrating, the usage of this function was limited to two positive and negative ratings each. A total of 18 positive and 17 negative out of at minimum 52 possible ratings for 13 games however shows that the limits have not been exhausted. Nevertheless, the ratings for this model are mostly correct, especially for misspelled or wrong annotations like “ufo”, “startwars”, “starwars” or “enterprise”. But beside correct ratings, there are also valid associations like “ship” or “propeller” that were downrated or invalid associations that were uprated. Additionally, both obvious and less obvious annotations, for example “boat” or “speed” were not rated at all. Thus it is hard to make a general statement about the quality or use of user ratings for this specific model.

When considering only the first annotation of each user it is salient that the distribution for the hovercraft-model is similar to the distribution when considering all entered annotations, at least for the top three. This is especially interesting as the hovercraft-model is one of only two models where a color has been entered more than once as the first annotation. The ratio of the first- and second-most entered annotations among those entered first is however higher than for all annotations. This leads to the conclusion that color is an obvious feature for humans if the model cannot be recognized or if no other well-known annotations for the model can be given.



(a) 3D-model of the “Thing”



(b) Annotations and user ratings for 3D-model the “Thing” with absolute number of occurrences in blue and number of occurences for first entered annotations in red

Figure 5.2: This figure shows a 3D-model of the “Thing” (a) with its top 20 annotations and user ratings (b).

5.1.2 Special Model

The second model to be analyzed is the "Thing" from "The Fantastic Four" shown in Figure 5.2a. Figure 5.2b shows a bar plot of the top 20 annotations with their absolute number of occurrences as well as the number of occurences when only considering the first entered annotations, like for the hovercraft model before. It shows that there is no clearly best annotation, but that the counts of the first four items are almost similar. A total of 62 annotations, containing 32 unique ones were collected, which is similar to the results described for the "hovercraft" model. But the top three annotations cover only about 31% of all given annotations. That is by far the lowest value of all models in the test set.

What is interesting is that the almost equally distributed top four items are separated into very specific and very general annotations. The former category consists of the exact name of the character and the related comic and the second one only contains the colors of the model shown. By comparison of those annotations with the first entered ones we can see that the color has only been entered once. The distribution shows that more specific annotations have been entered first, while the color seems to be entered additionally when no more specific annotations were known. Likewise, for models like the elephant, duck, butterfly or the bicycle where color is a dominant annotation in general, those entered first show a different distribution. It thus seems that the annotations entered first gives an insight into which features are especially relevant to humans.

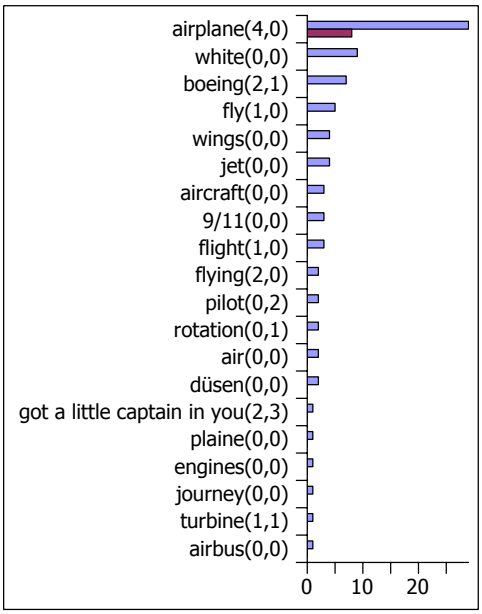
Even if the quality of the model shown is good, it is somehow special. Not everybody is familiar with Marvel comics or the screen adaptation and therefore may not know the character. It seems that in this case, users tend to enter the most common terms possible to keep the chance of matching other players keywords. When comparing the results of the "Thing" and "Bumblebee" from "Transformers" in Appendix A, it is salient that the results of Bumblebee are much better in terms of absolute count. A concrete proposition about the reasons for these differences cannot be given, but a guess is the higher popularity of "Transformers" compared to "The Fantastic Four" due to more frequent representations in movies in the last years. But when considering only the first annotations entered, we can see that for both the model of the "Thing" and the "Transformer" the top-annotation matches exactly the title of the movie they are known for.

The user ratings for this model are mainly correct, except the downrating of orange. But as for the model described before, the rating feature was also used very little. With a total of 13 up- and 5 downratings for at least 48 possible ratings in 12 games, the participation is even worse than for the first model described.

5.1.3 Common Model

The last model to be described in more detail is an airplane shown in Figure 5.3a. With 29 occurrences out of a total of 96 entered keywords, its top annotation "airplane" has the highest number of matches from all models in the test set. It covers 31% and the top three annotations cover 48% of all annotations entered. As for the models before, the bar plot for annotations and the number of occurrences is shown in Figure 5.3b. While the ratio of 96 entered keywords to 30 unique ones is relatively high and the drop from the best to the second best annotation is high

(a) 3D-model of an airplane



(b) Annotations and user ratings for 3D-model "airplane" with absolute number of occurrences in blue and number of occurences for first entered annotations in red

Figure 5.3: This figure shows a 3D-model of an airplane (a) with its top 20 annotations and user ratings (b).

too, the coverage of the top three annotations as well as the number of distinct ones are similar to those of other models. So the number of annotations entered by a player may depend on how special a model is. The higher the number of well-known terms is for a specific model, the more likely it was that almost equally distributed annotations have been given. For example the model of a bicycle shows similar counts for "bike" and "bicycle" or the model of headphones show high counts for both "headphones" and "earphones" as well as "sofa" or "couch" for the furniture model. But the number of matches for annotations other than the best ones seems to decrease if a model is more specific. Duck, airplane, hovercraft, astronaut or butterfly are examples for this. A general model of a bird may lead to more equally distributed counts, while a model of a penguin may shows a more unique peak.

The distribution of the annotations entered first confirms this hypothesis. Except for one user for "737", all users entered "airplane" first. This also holds for other common models like the butterfly or the elephant.

The user ratings for the airplane are good most of the time and discard annotations like "pilot" or "got a little captain in you". With 15 to 12 positive ratings in 13 game rounds, the possible ratings of at least 52 have not been reached by far. This follows the trend of a low participation in rating other players words.

5.1.4 Comparison of 3D-Model Data in General

When comparing the results of all models in the test set, it is obvious that for every model at least one color is inside the top five annotations. Except for three models, a color is even inside the top three. This leads to the conclusion that color is either a important model feature for users or color acts as a substitution if a player does not know any specific annotations. The number of matches for entered annotations seems to depend on how special models are. The more equivalent terms exist for a model, the more evenly distributed the counts for the top annotations are. In contrast to that, models which do not have multiple equivalent terms show a distinct peak for the best annotation. However, when considering only the first-mentioned annotations, it can be seen that color becomes less important while more specific annotations show higher peaks. For all models in the test set those annotations entered first match exactly the model shown. So by taking into account not only the overall number of occurences, but also the count for those entered first, it is possible to separate specific annotations from more general like color. A promising approach is therefore to rank the resulting annotations not only by their total number of occurences, but also by their number of occurences when considering only the first input of users. This way, specific annotations can be collected and complemented by descriptive ones like color.

The user rating feature has not been used to the possible degree, so increasing the allowed positive or negative ratings should not have an effect on the collected data. Furthermore, the two positive ratings for the airplane annotation "got a little captain in you" show that the system is not invulnerable to wrong ratings. Even if the users were told to only give true ratings, bad but funny annotations were uprated and plausible ones were downrated. This inconsistency confirms the statement from Theodosiou and Tsapatsoulis in [32] that annotations are strongly dependent on the users view in a specific context and time. Because of the low participation, it is impossible to say if the results of the ratings would have been correct if the number of ratings collected would have been significantly higher. Altogether, it still could be an option to let user ratings have some influence on annotations with a low number of matches. For the top annotations this should not be necessary, as the results prove that the top annotations represent the model shown to a high degree.

Each model is well described by its top three annotations and the top five contain only valid terms, attributes or associations but no apparent wrong annotation. Furthermore, it does not seem to be a problem if models are of lower quality, because the human brain is well trained in recognizing 2D images as well as 3D objects. As long as the most important features of the models can be recognized, the results are expected to be similar to those of the user study. Using the top three annotations and complementing them with ones that have a lower number of matches but positive ratings could be a good way to get a larger set of annotations for each model. Nevertheless, it is advisable to double-check these complementary annotations with other mechanics to prove correctness. This is briefly discussed in Chapter 6.

Like for the three models shown above, bar plots for all models can be seen in Appendix A. They show the top 10 annotations as well as the number of occurrences per annotation. Table 5.1 furthermore shows a list for all models which contains the number of unique annotations, the coverage of the top three annotations and the percentage of users that entered the best annotation.

	# unique Items	Top 3 Coverage	% Top Annotation
Rabbit	20	51.16%	83.33%
Thing	32	30.56%	58.33%
Duck	29	47.62%	100.00%
Airplane	30	48.39%	100.00%
Elephant	27	46.67%	91.67%
Bicycle	31	47.62%	66.67%
Butterfly	18	50.00%	100.00%
Gorilla	30	46.07%	82.35%
Hovercraft	31	39.71%	64.71%
Teapot	29	47.37%	87.5%
Transformer	26	52.00%	92.86%
Headphones	24	52.83%	83.33%
Astronaut	37	44.05%	100.00%
Couch	30	42.62%	84.62%
Camera	21	42.42%	81.81%
Car	30	42.50%	85.71%
Saw	36	47.06%	88.24%
Screw	22	53.13%.	100.00%
Screwdriver	25	45.00%	81.81%

Table 5.1: Statistics for the data collected within the model annotation round. The table shows the number of distinct annotations, the percentage of covered total annotations by the top three and the percentage of users that entered the annotation with the highest number of matches.

5.2 Analysis of Land Cover Data

We analyze three different land cover maps in detail in this subsection, including a map with low resolution, one with a higher resolution but a variety of different land cover types and a homogenous, high-resolution map with well-known features. This exemplarily shows the results of the user study which includes 20 different land cover maps with varying resolution and distribution of land covers. We intentionally added maps of different quality to the test set to examine if crowdsourcing is effective for improving land cover data obtained by remote sensing. Thus for example some maps with very low resolution are also added as well as a map from the ocean which just shows a black rectangle. Furthermore, maps with medium resolution but with multiple land covers as well as maps centered at popular points of interest like the Pyramids of Giza, Palm Island in Dubai or the Colosseum in Rome are used.

5.2.1 Low-Resolution Map

First, a low-resolution map is discussed which shows an area of uncultivated soil somewhere in a mountainous region. The user is presented a zoomed section of the map and the area of interest is marked with a red square, centered on coordinates that are of interest for the Geo-

Wiki project [12]. The coordinates from the project usually define regions where different land cover maps show large differences as already described in Chapter 1. The map can be seen in Figure 5.4. The top 20 annotations are plotted in Figure 5.5a and show that the most frequently entered annotation is “green”, followed by “fields”, “shrubs” and “brown”. While the top four annotations are not incorrect, they do not give much information about the underlying land cover except for “fields”. But identifying the land cover correctly is hard for this map, because the low quality does not allow to distinguish between grass, shrubs, uncultivated soil or maybe trees. The only unique hint for cultivated soil are the rectangular parcels of land. But without further context, it is hard to annotate the map correctly.

Comparing annotation results with the results from user drawings presented as a box plot in Figure 5.5b shows that due to the pre-defined categories, users were mostly able to correctly categorize the underlying land cover type. The representation as box plots were chosen, because it is a suitable graphical description of the distribution and dispersion of cardinal scaled data. It shows five robust measures of dispersion in one plot, namely the lower and upper quartiles, median and the sample minimum and maximum. Lower and upper quartile, also called 25th and 75th percentile, splits the lowest and respectively highest 25% of data. The interquartile range (IQR) is the range between the upper and the lower quartile and represents the data’s middle 50%. The length of the box equals the IQR and is bounded by the lower and upper quartile. The median which cuts the data into a lower and upper half is shown as a horizontal line inside the box. The whiskers show outliers which are in the range of the one and a half of the IQR and extreme outliers are represented as circles.

Here, too, the user rating feature was hardly used. 15 positive and 6 negative out of a total of at least 48 possible ratings in 12 games show need for improvement, even if the downrating of bad or annotations in other languages than english worked very well.

A total of 14 drawings were collected for this map and the box shows the distribution and dispersion of these data. In this analysis, we are going to neglect outliers and focus on the mainly chosen categories. All categories other than “shrub” and “cultivated soil” are therefore negligible, so we take a more detailed look at these two. The box for shrubland shows an IQR from zero to 0.14 with the median at 0.0027. This means that 50% of the users chose almost no pixel to be shrub and the other 50% painted between 0.3% and 14% of the map with the shrub icon. Two users categorized about 100% and one about 70% of the map as shrubland, but these are outliers as visible in the plot. The IQR for cultivated soil ranges from 0.12 to 0.97 with the median at 0.77. So 50% of all users categorize more than or equal to 77% of the map as cultivated soil. This is a good result considering the low quality of the map and matches the top two annotations given. By combining annotations and user drawings, satisfactory results can be achieved even for maps where annotations alone are not sufficient.

5.2.2 Medium-Resolution Map

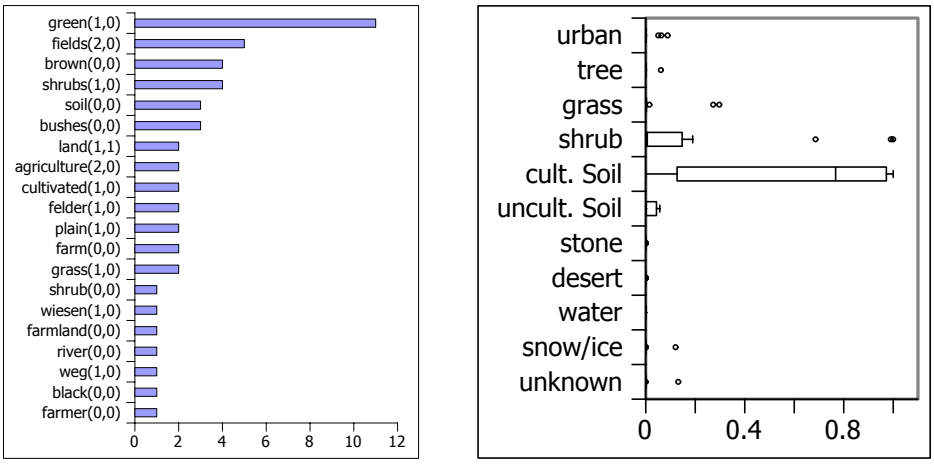
The second map can be seen in Figure 5.6 and shows a region near the summit of a mountain situated nearby the area of the first map. Even if coordinates for both the map described before and this map only differ minimally, the land cover types do completely. We already saw that the first map shows cultivated soil, but the second map represents an area of stone, snow and ice. The map is of a medium resolution but the different types of land cover can be well distinguished.

Figure 5.4: Low resolution map of an area of cultivated soil

This is reflected in the annotations given by the users. Figure 5.7a presents all annotations with the corresponding user ratings and the total of matches for each category. Here, all annotations are shown because users only entered 19 distinct ones. The bar plot makes it clearly evident that all of the top seven annotations match the land cover types on the map. Together, they cover about 84% of all annotations given for this specific map, which is an almost perfect result.

Taking a look at the box plot in Figure 5.7b reveals a distribution that supports the top annotations given by the users. The IQR for water ranges from 0.02 to 0.033, which matches the percentage of water on the map nearly exactly, although because of the lack of ground truth data, it cannot be said if the area centered on the top consists of water or ice. For the category of snow or ice, the lower quartile is at 0.13 and the upper at 0.27 with a median of 0.21. This is also a good estimation of the actual area covered by snow or ice. Primarily, the map was marked as stone, with a lower quartile at 0.44, the upper at 0.65 and the median at 0.60.

Altogether, these results for land cover categorization are very good, as it is nearly impossible to paint the map 100% correct. This is because the borders of stone and snow/ice are blurred like for stone semitransparently covered with ice. This map proves that both annotating and drawing on maps with medium resolution can yield expressive and correct results as long as the different land cover types can be recognized and distinguished from each other. The user ratings are mostly correct, especially when comparing positive to negative ratings. With 22 positive to 15 negative ratings in 12 rounds, the participation was a little higher compared to other maps, but they concentrate mainly on the top 7 annotations. Like already said, it cannot be decided if the rating for “water” is good or bad, as we do not know the ground truth of the elliptical, blue region centered on the top.



(a) Top 20 Annotations (b) Drawing Results

Figure 5.5: This figure shows the top 20 annotations with absolute number of occurrences and user ratings (a) and the drawing results from all user drawings (b) for the map in Figure 5.4.

5.2.3 High-Resolution Map

The last map that is examined in detail shows one of the three Palm Islands in Dubai, called "Jumeirah" and can be found in Figure 5.8. It is an artificially built island in form of a palm. It lenghtens the shoreline of Dubai by about 100 kilometers and is a well known sight. This map has been chosen to test if users tend to describe the land cover or to enter names for objects they can recognize. In addition, maps centered at the Pentagon or the Pyramids of Giza as well as Venice or the Colosseum in Rome are contained in the test set. Their results can be found in Appendix A and B.

The bar plot in Figure 5.9a plots the top 20 annotations with their corresponding user ratings and their absolute number of occurrences. Out of 33 distinct annotations, the best seven are all correct and cover 64% of the total of annotations entered. All of them are confirmed by the positive user ratings. Only one negative user rating for "island" is incorrect. The participation of about 50% is again low compared to the possible number of ratings, but the quality is good. Before we compare the results of the game-rounds for annotation and categorization, it should be said that this map is hard to paint because of the relatively fine structures and alternating categories.

Figure 5.7b presents a boxplot of the results that were collected in the drawing round. Except for outliers, all users painted the map using only the categories "urban" and "water". This nicely matches the land cover types the map is covered with. The urban's lower quartile is 0.44, the upper quartile is 0.61 and the median is 0.53. For the water category, the lower quartile is 0.12, the upper is 0.39 and the median is 0.31. Depending on the brush size and the resulting granularity, the results shown are near the optimum and reflect the ratio of water and urban area well. Even if the results for annotations and drawings are separately good, it is hard to combine them because users may not necessarily enter land cover types but tend to enter names of objects

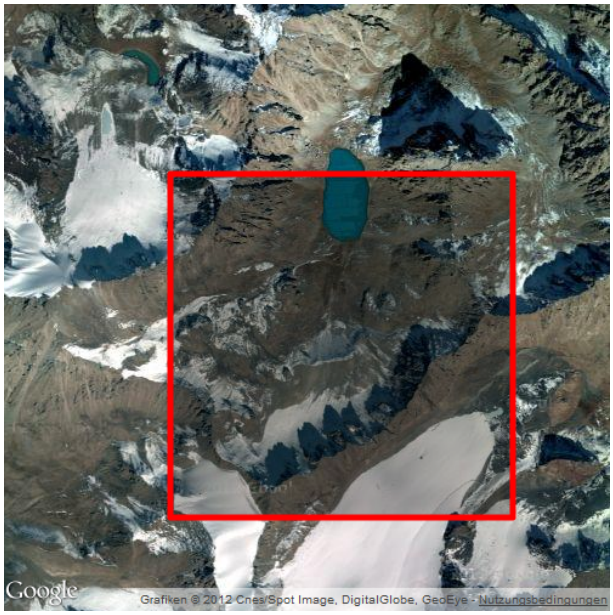


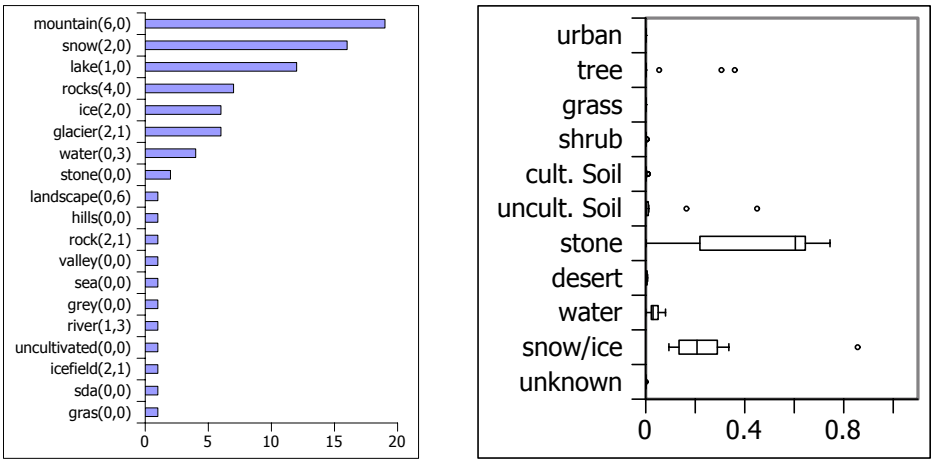
Figure 5.6: Medium resolution map of an peaky area of stone, snow and ice

and sights they see. By collecting user annotations and paintings apart from each other, like the game does, it is however possible to define the different land cover types and to additionally collect meta-information like object names, location names and others.

5.2.4 Comparison of Land Cover Data in General

The data collected by the second and third round of the game prove, that it is possible to refine land cover data by using a crowdsurcing game with user annotations and drawings. 20 different maps were used in the user study. Each map of medium or even good resolution shows meaningful and matching annotations. Only those with low resolution led to more common annotations like colors or wrong annotations like shrubs instead of cultivated soil as for the first map shown in this subsection. The reason for that is the lack of details in the map. Thus users may not recognize or distinguish different land cover types.

Additionally, ambiguous land cover like vegetated or ice-covered stone led users to choose the more succinct one. The collected user drawings show excellent results, which can be well visualized using box plots. By using robust features like the median instead of the mean, outliers can be identified and do not have as much effect on the data as the mean value would. Although both annotation and drawing rounds aim at the collection of different data, the combination of both promise satisfying results. Barplots for user annotations as well as boxplots for user drawings can be found in Appendix B and C. It should be said that in some cases, the box does not show the median. This is because it is near zero and thus plotted on the x-axis.



(a) Top 20 Annotations (b) Drawing Results

Figure 5.7: This figure shows the top 20 annotations with absolute number of occurrences and user ratings (a) and the drawing results from all user drawings (b) for the map in Figure 5.6.

New gathered annotations Here, we briefly show annotations that have been collected within the annotation round for land cover. In general, the number and quality of annotations seem to depend on the number of different terms users know for a given type of land cover. For example for urban areas more good as well as bad annotations have been collected. In contrast to that there were only four distinct annotations for grass and only one useful annotation for "uncultivated soil", which is exactly the category name used in the game-round for land cover categorization. The number of annotations admittedly depends also on the area covered with a given land cover for the maps used in the test set.

Like for the model results above users tend to enter very specific annotations for well recognizable land cover like "river"or "lake" as well as "houses" and "streets". It has been already mentioned that also objects have been labeled, like the pyramids or the Pentagon. To extract the most promising but also ambiguous annotations we manually assigned all entered annotations to the categories used in the land cover categorization rounds. At the same time annotations that cannot be assigned to a unique category have been discarded. These include annotations like "landscape", "plain", "island", "Dubai", "pyramids" and many others.

A list of promising annotations ordered by the number of occurrences can be found in Table 5.2. It can be seen that the only valid annotation for the category of uncultivated soil is "uncultivated soil" itself. The reason for this seems to be that users did not know any further correct term. It seems furthermore natural that the results are biased, because players seem to get used to the terms which are used for the selection of categories in the drawing round. The result is that the category name itself is contained in the list of valid annotations for each category. For the category "unknown" it is salient that the annotations belong to the map which shows only very dark sea. Except for this map players did not use any paraphrase of "unknown" for any map. Instead, if players are not sure about the type of land cover shown on the map they

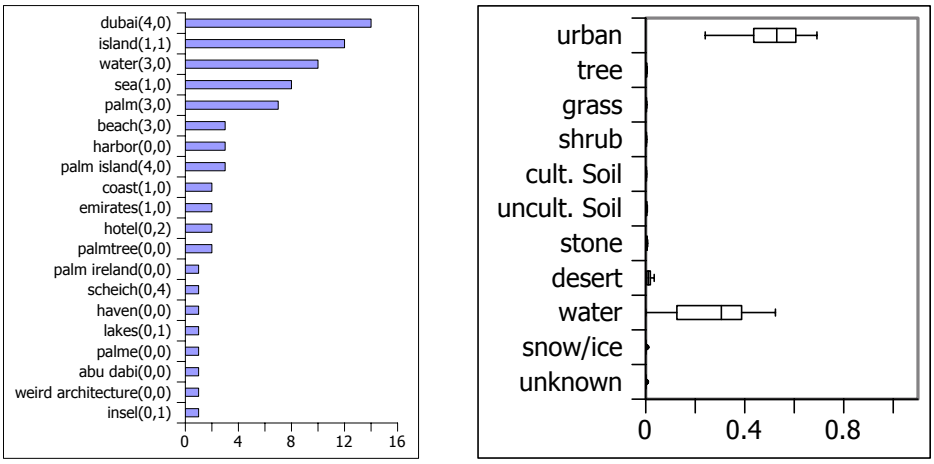
Figure 5.8: High resolution map of one of Dubai's Palm Islands

Urban	city, urban, town, houses, streets
Tree	trees, woods, forest
Grass	grass, grassland, meadows
Shrub	shrubs, bushes
Cult. Soil	cultivated soil, fields, soil, agriculture, farmland
Uncult. Soil	uncultivated soil
Stone	rocks, mountain, hills, stone, ridges
Desert	desert, sand, dunes
Water	water, river, lake, sea, ocean
Snow/Ice	snow, ice, icefield
Unknown	unknown, undefined, nothing, darkness

Table 5.2: Promising annotations collected for land cover maps

tend to provide annotations for the assumed type of land cover. Especially for stone a couple of annotations could be found that occured often like "rocks", "mountain" or "hills".

Table 5.3 shows annotations that have been assigned to one of the given categories but did not match very well. The reasons for that are versatile. The keywords "london" or "bridge" do match the map well that shows London and the Thames , but are too specific to be used as general annotation for urban areas. Likewise are colors like "green" for the categories "tree" and "shrub" ambiguous and could match grassland or cultivated soil as well. This carries on for annotations like "valley" or "alps" for stone, "sahara" for desert as well as for "atlantic" or "blue" for water.



(a) Top 20 Annotations

(b) Drawing Results

Figure 5.9: This figure shows the top 20 annotations with absolute number of occurrences and user ratings (a) and the drawing results from all user drawings (b) for the map in Figure 5.8.

Urban	bridge, london, industry, hotels, palm island, pentagon
Tree	green, park
Grass	—
Shrub	green
Cult. Soil	crops, crop circles, corn
Uncult. Soil	brown, earth
Stone	valley, brown, glacier, alps, high
Desert	sandy land, brown, sahara
Water	blue, coast, beach, atlantic, pond, thames
Snow/Ice	—
Unknown	void, empty, black hole

Table 5.3: Bad, too special or ambiguous annotations collected for land cover maps

Usability of New Annotations We showed before which new annotations have been extracted by letting users annotate global land cover maps. It is hard to give a general statement about the usability of those annotations. It depends on the type of land cover if and how well those new annotations are applicable to be used as sub-categories. There are annotations almost equivalent to each other like “woods” and “forest”, which could be used as replacements to summarize those annotations. But for maps with a very high resolution a single tree would not match the subcategory “forest”. Others like “grass” or “meadows” as well as “ shrubs” or “bushes” are interchangeable. Annotations like “river”, “lake” or “sea” however give the opportunity to describe a given type of land cover more precisely. Likewise urban areas can be sub-categorized in houses or streets. This however requires maps with a resolution high enough to clearly distin-

guish streets from houses. Also rocks could be generally distinguished from mountains or hills. It is however questionable if non-expert users are able to assign those sub-categories correctly, especially if only a small area of the map is shown without any overall context.

Extracting sub-categories could potentially give the opportunity to refine the categorization of land cover maps. It is still advisable to use caution and to double-check the gathered data to avoid wrong assignments. Using those sub-categories for painting on maps could be useful. It might be good to show specific maps with a limited number of different types of land cover to the users. Then, sub-categories could be used to get more precise information about land cover. For drawing on maps providing all the categories used in the user-study and also sub-categories might overstrain users and might lead to a decreasing quality of the resulting data. On the other hand, more specific annotations could be summarized to the general categories used in the drawing game-round by using sub-categories gathered by user annotations and assign them to the matching general category.

5.3 General Insights

In general, it seems to be proven that annotating 3D-models is a task which is easy to solve for humans, as long as the shape can be recognized and associated with known objects. From all 19 models contained in the test set, at least the top three annotations match the model very well. An important feature for users is color, which is present in all results and mostly inside the top three annotations. Especially for more special models like the “Thing” or a hovercraft, color seems to be a substitution for specific annotations. The reason for that could be that users are unfamiliar with the model. Also the model of a duck shows a high peak for “yellow”, but that should be because the color is very dominant and glaring and captures the focus of the players. By considering only the first-mentioned annotation for each user, color become less important. The top-annotations of those entered first are mainly very specific and match the models shown very well. However, by combining both results, specific annotations can be found and complemented with descriptive ones like color.

The distribution of the frequency of the top annotations seems to depend on how many correct terms exist for a given object. For example, the frequency of “rabbit” and “bunny”, “headphones” and “earphones” as well as for “sofa” and “couch” are rather equally distributed, while models with rather unique terms like “airplane” and “screw” show more distinct peaks. Also associations like “hot” for the teapot or “flying” for the butterfly model were entered. They are outside the top annotations but could be used as additional information. The user rating feature was not used to its possible extent, although the majority of ratings is correct and could be helpful to further distinguish good and bad annotations.

The annotation of land cover maps achieved good results for maps with a resolution that is high enough to recognize and distinguish land cover types. For low resolution maps, the majority of annotations are not wrong, but do not deliver meaningful terms. That is because it is necessary to identify and dedicate the structure and color of a given land cover to be able to correctly annotate it. Without that, only assumptions based on the surrounding or only the color are possible. This leads to wrong guesses like for the map shown in Figure 5.4.

The vast majority of maps with a higher resolution however delivered good and correct results, although the annotations are sometimes very special. For example, Palm Island in Dubai was recognized as well as the Pyramids of Giza or the Colosseum. This is not bad per se, but makes it more difficult to merge the results of annotations and drawings. In addition to this, annotations cannot be used to describe proportions and the absolut number of occurrences does not necessarily give information on the land cover distribution of a map. On the other hand, allowing the user to paint on land cover maps to categorize different land cover types is perfect for doing this. The results are excellent and reflect the land cover distribution to a high degree. 18 out of 20 maps were definitively categorized correctly. For the two remaining maps with monotonic and ambiguous land cover, the category that covers the biggest areas was classified correctly. The distribution of the categories with the second and third highest frequency however looks similar. This means that users were not able to uniquely identify the land cover type correctly.

Because of the pre-defined categories, users are however limited and can only use categories that are available. If no category matches a specific land cover type, the next best category has to be used instead. But for the sake of usability and ease, having a large number of different categories to choose from is ineligible.

CHAPTER 6

Conclusion and Future Work

We presented mechanics to collect meaningful data for both 3D-models and land cover maps by exploiting the potential of crowdsourcing combined with social games. The rising number of people connected to the internet and the popularity of social games and platforms like Facebook gives the opportunity to reach a large user base and to channel the abilities of non-expert users to solve useful tasks. We created a game that fulfills the most important heuristics for social games and gives incentives to users to spend their spare time while providing valuable data for different research areas. We focused our attention on not to exclude users because of missing support for plugins or external players and thus implemented the game using only HTML5 and JavaScript. It is supported by all major browsers including those of nowaday's tablets and smartphones. In a preprocessing step, models are rendered into videos to remove the dependencies on WebGL, which is not supported by mobile devices today. The game consists of three game-rounds, where each round tackles another problem. This includes the collection of 3D model annotations as well as land cover annotations and user drawings to categorize different types of land cover.

The results gathered by the user study described in Chapter 5 prove that humans are able to provide good annotations for 3D models, as long as the quality and resolution is sufficient to recognize shape and relevant features, like structure or color. For land cover maps, the results show that the significance of annotations corresponds to the resolution of the map and the distinguishability of the different types of land cover. Users furthermore tend to name objects or sights they recognize, like Palm Island or the Pyramids of Giza. This gives additional informations, but makes it harder to compare the results achieved with annotations and drawings. While the annotation of maps gets more rambling results, drawing a map with pre-defined categories does not. Instead, the results prove the ability of humans to categorize land cover correctly, even if the corresponding map is of low resolution. All maps used in the test set were classified correctly and the annotations match the categorization to a high degree. The combination of drawings and annotations thus offers the possibility to collect a variety of data.

The results achieved by the project can be used to refine annotations for model databases and thus improve retrieval. Additionally, the results can be used as a training set for the development of automatic algorithms for model recognition. Furthermore, the results for land cover maps may

help scientists in the area of climate change, land use or ecosystems and support the ongoing projects *Geo-Wiki* [12] and *Landspotting* [2].

Future Work

As we could show, our game can deliver good results for arbitrary land cover maps and 3D-models. The only constraint is that the quality of maps and models needs to be acceptable. Otherwise, humans are not able to provide useful information. The quality of many maps available from Google Maps is however very low. This includes images that are, for example, black, covered with clouds or are just blurry or pixelated due to the lack of resolution. To get useful results for arbitrary maps, it is thus also necessary to improve the quality of the underlying data.

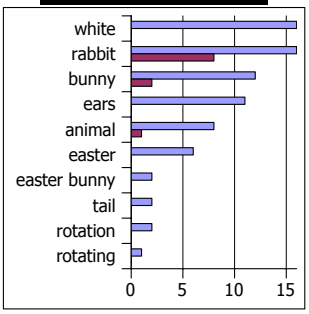
By the analysis of the collected data, we also realized that the user rating-feature was not used to the possible extent. The possibility of validating annotations with a low frequency is therefore not always possible. A solution could be to include a game round, where annotations need to get assigned to models or maps. That way, the annotations could be double-checked to a considerable degree while keeping up the game character. Enabling the users to annotate land cover maps shows good results, but it is however not possible to make a point about the location of the described features. By displaying markers for specific points of interest, annotations could be assigned to spatial coordinates. The drawback however is, that this can only be used for certain specific points, because annotating whole maps with this approach would need a vast number of users.

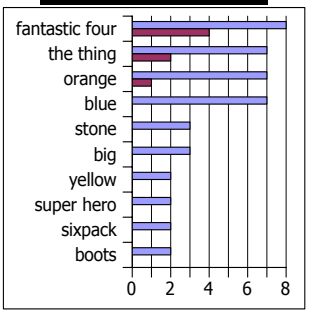
Another problem to solve is that terms like “shrubs” or “cultivated soil” may not be familiar to users with a first language other than english. The vocabulary thus influences the quality of annotations. Ideally, every user could enter annotations in his first language, without having any disadvantages caused by game mechanics or scoring. It may further improve the results if user inputs would be checked by using dictionaries and stored in a consistent language. This is because the potential number of users would grow as well as the number of annotations a user can provide. At the same time, misspelling or typing errors could be handled with this approach.

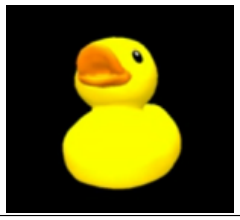
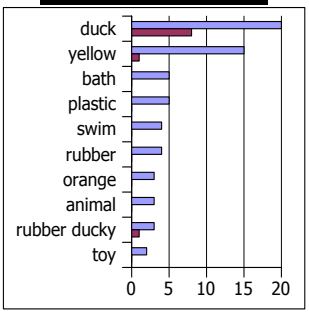
APPENDIX A

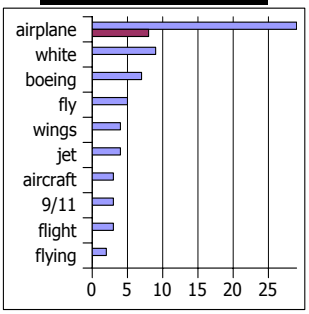
All Results of Model Annotations

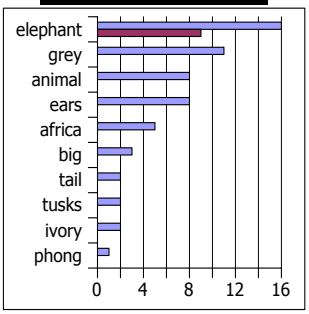
This appendix shows bar plots for all models in the test set, where the x-axis shows the top ten annotations entered by users. The y-axis shows the absolute number of occurences for each of the top annotations. The blue bars indicate the total number of occurences for all user-annotations and the red bars show the number of occurrences, considering only the first entered annotation for each user.

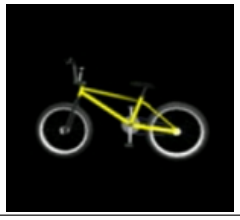
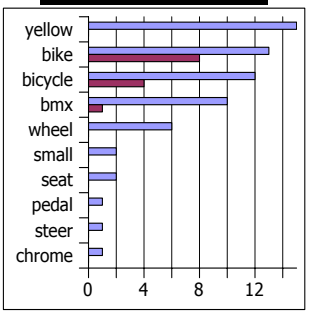


white
rabbit
bunny
ears
animal
easter
easter bunny
tail
rotation
rotating
0
5
10
15

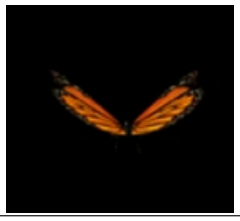
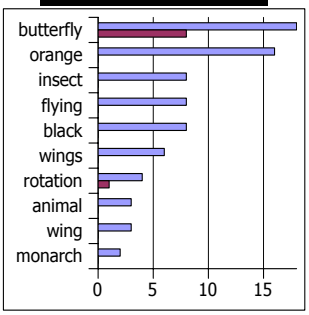


fantastic four
the thing
orange
blue
stone
big
yellow
super hero
sixpack
boots
0
2
4
6
8

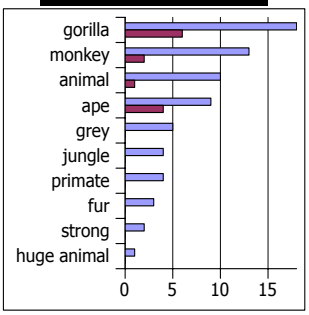


duck
yellow
bath
plastic
swim
rubber
orange
animal
rubber ducky
toy
0
5
10
15
20

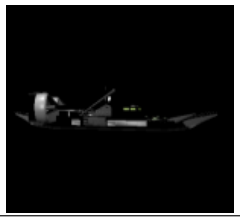
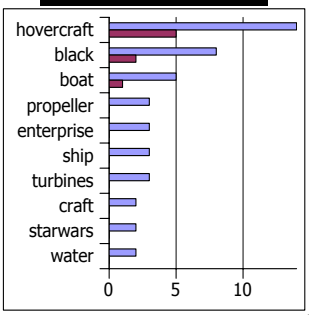


airplane
white
boeing
fly
wings
jet
aircraft
9/11
flight
flying
0
5
10
15
20
25

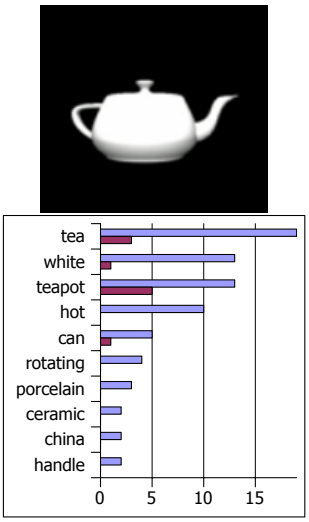


elephant
grey
animal
ears
africa
big
tail
tusks
ivory
phong
0
4
8
12
16

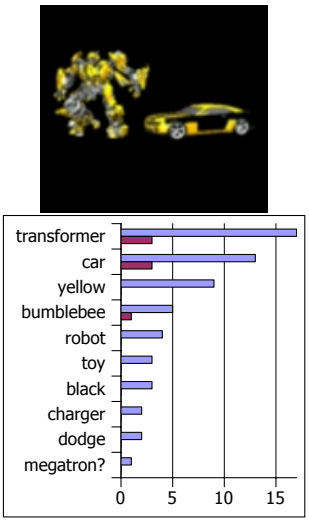


yellow
bike
bicycle
bmx
wheel
small
seat
pedal
steer
chrome
0
4
8
12

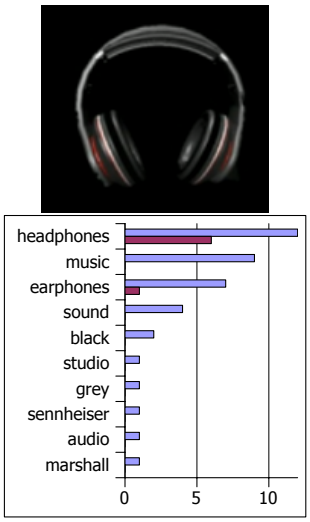


butterfly
orange
insect
flying
black
wings
rotation
animal
wing
monarch
0
5
10
15

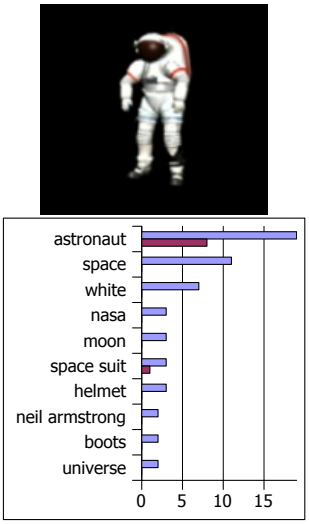


gorilla
monkey
animal
ape
grey
jungle
primate
fur
strong
huge animal
0
5
10
15

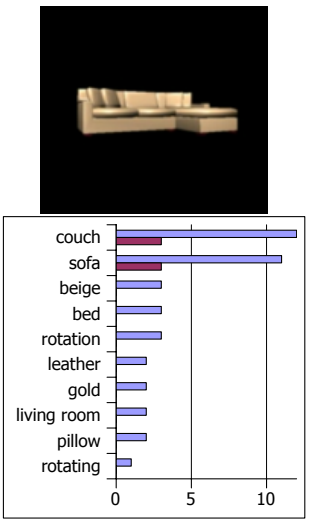


hovercraft
black
boat
propeller
enterprise
ship
turbines
craft
starwars
water
0
5
10

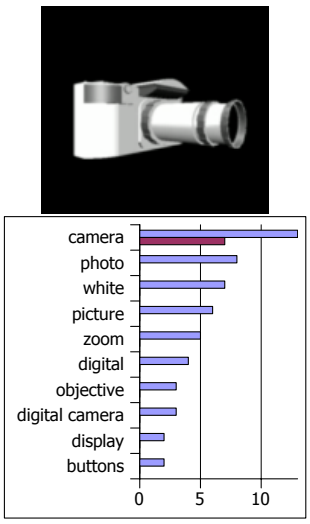

tea
white
teapot
hot
can
rotating
porcelain
ceramic
china
handle
0
5
10
15

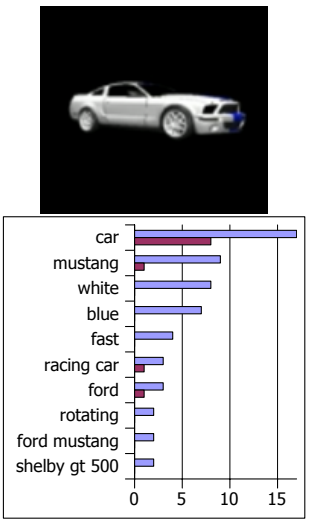

transformer
car
yellow
bumblebee
robot
toy
black
charger
dodge
megatron?
0
5
10
15

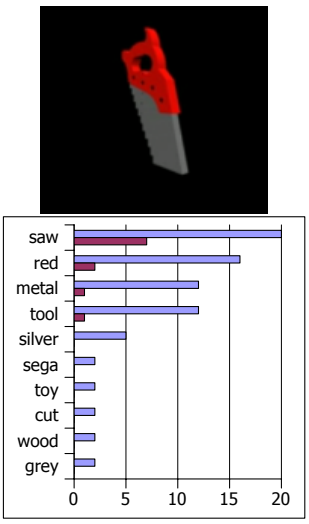

headphones
music
earphones
sound
black
studio
grey
sennheiser
audio
marshall
0
5
10

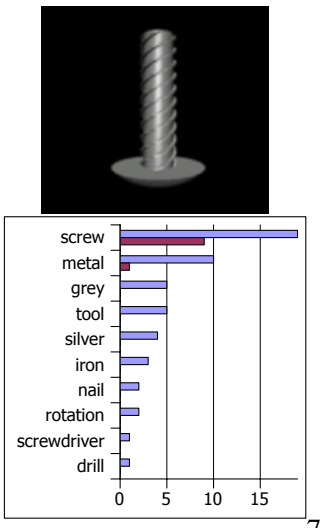

astronaut
space
white
nasa
moon
space suit
helmet
neil armstrong
boots
universe
0
5
10
15

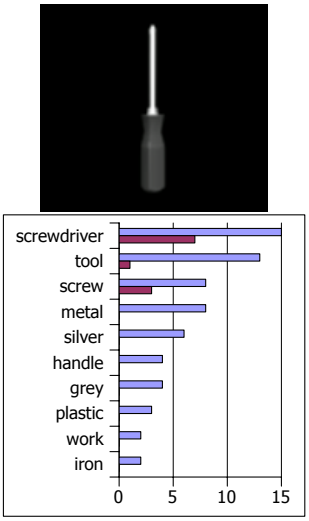

couch
sofa
beige
bed
rotation
leather
gold
living room
pillow
rotating
0
5
10


camera
photo
white
picture
zoom
digital
objective
digital camera
display
buttons
0
5
10


car
mustang
white
blue
fast
racing car
ford
rotating
ford mustang
shelby gt 500
0
5
10
15


saw
red
metal
tool
silver
sega
toy
cut
wood
grey
0
5
10
15
20

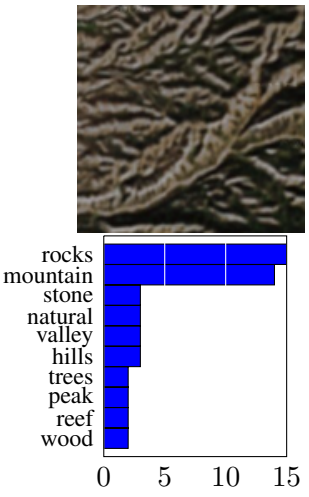

screw
metal
grey
tool
silver
iron
nail
rotation
screwdriver
drill
0
5
10
15

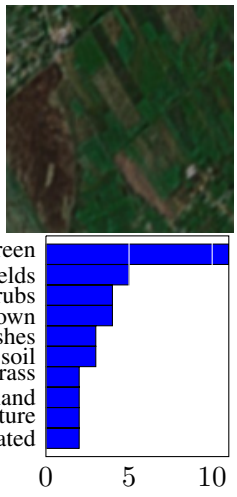

screwdriver
tool
screw
metal
silver
handle
grey
plastic
work
iron
0
5
10
15

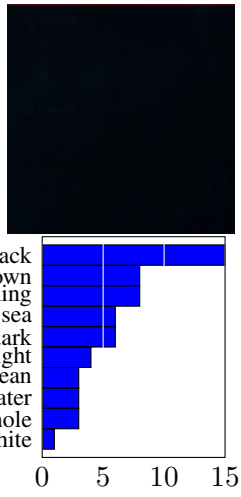
APPENDIX B

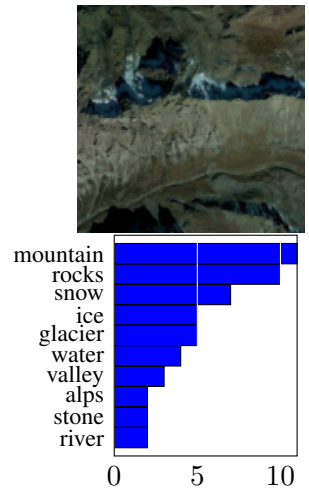
All Results of Landcover Annotations

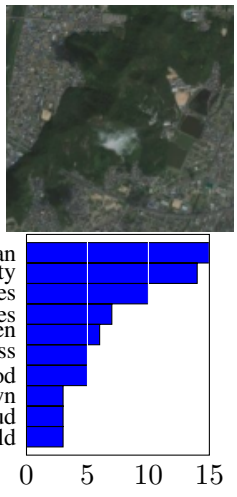
This appendix shows bar plots for all landcover maps in the test set, where the x-axis shows the top ten annotations entered by users. The y-axis shows the absolute number of occurences for each of the top annotations.

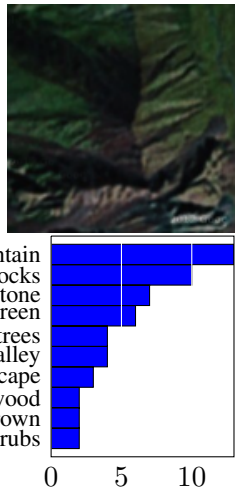

rocks
mountain
stone
natural
valley
hills
trees
peak
reef
wood
0
5
10
15

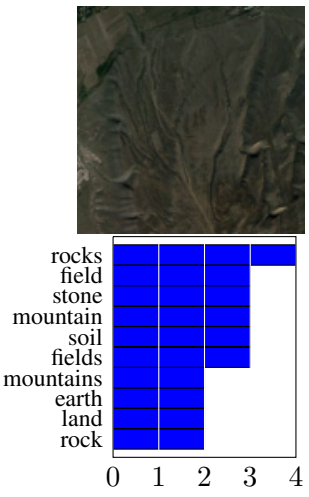

green
fields
shrubs
brown
bushes
soil
grass
land
agriculture
cultivated
0
5
10

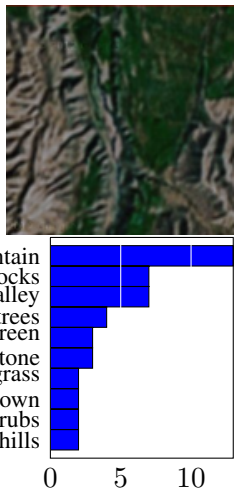

black
unknown
nothing
sea
dark
night
ocean
water
black hole
not white
0
5
10
15

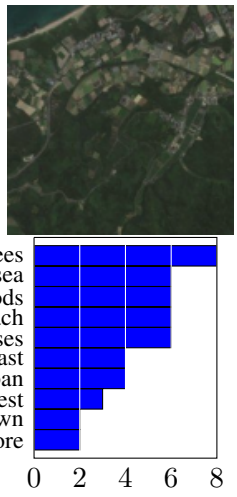

mountain
rocks
snow
ice
glacier
water
valley
alps
stone
river
0
5
10

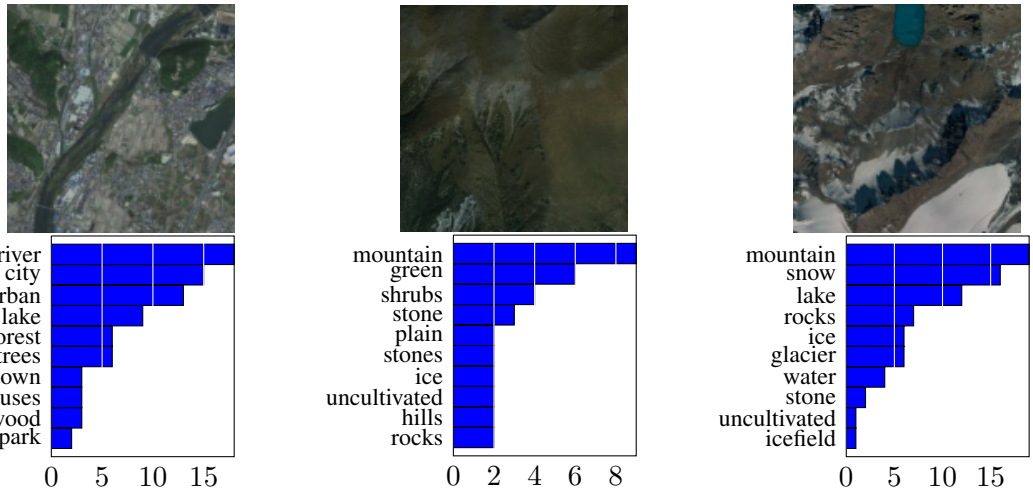

urban
city
trees
houses
green
grass
wood
town
cloud
field
0
5
10
15

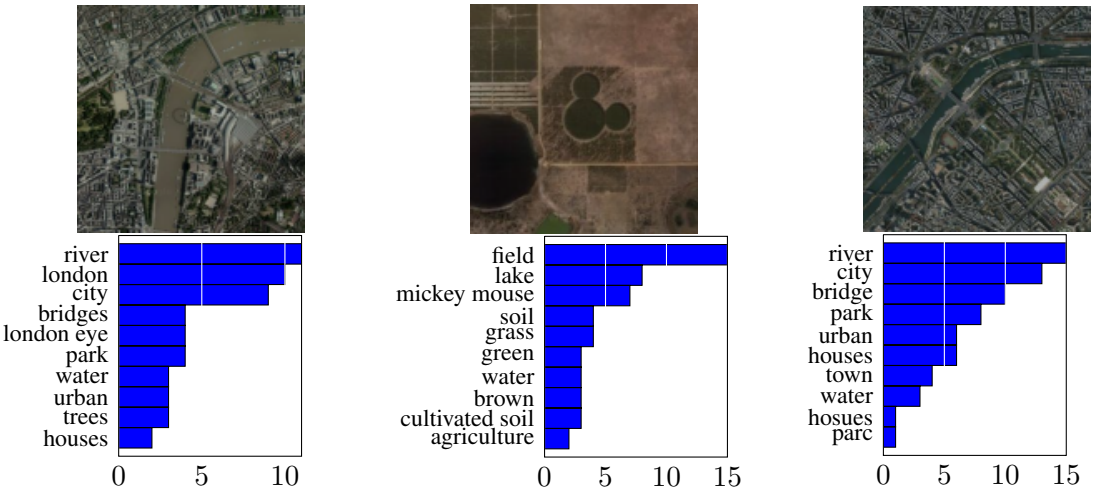

mountain
rocks
stone
green
trees
valley
landscape
wood
brown
shrubs
0
5
10

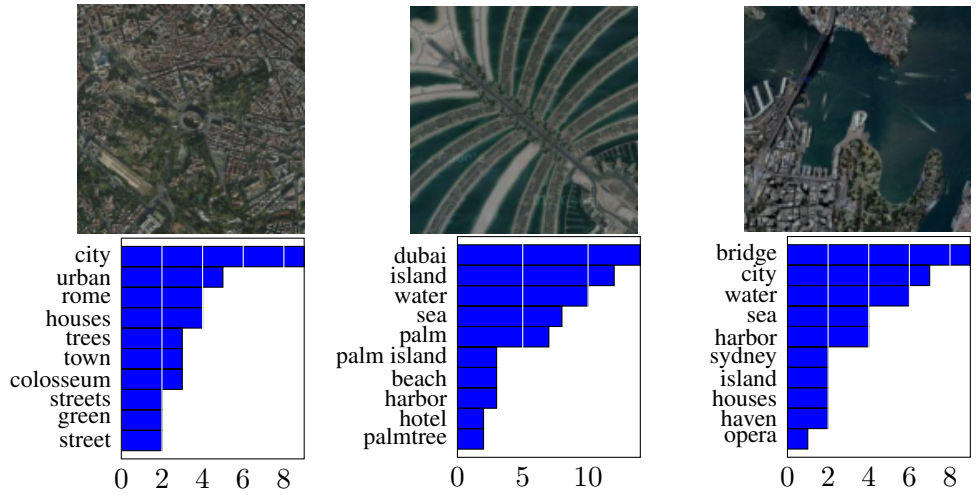

rocks
field
stone
mountain
soil
fields
mountains
earth
land
rock
0
1
2
3
4

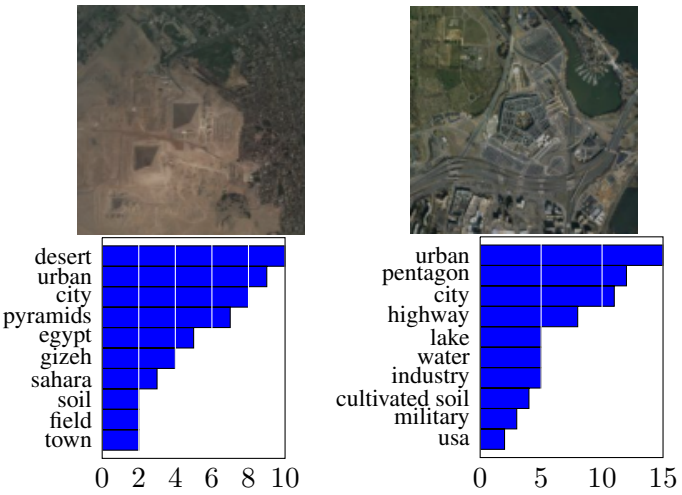

mountain
rocks
valley
trees
green
stone
grass
brown
shrubs
hills
0
5
10


trees
sea
woods
beach
houses
coast
urban
forest
town
shore
0
2
4
6
8


river
city
urban
lake
forest
trees
town
houses
wood
park
0 5 10 15
mountain
green
shrubs
stone
plain
stones
ice
uncultivated
hills
rocks
0 2 4 6 8
mountain
snow
lake
rocks
ice
glacier
water
stone
uncultivated
icefield
0 5 10 15


river
london
city
bridges
london eye
park
water
urban
trees
houses
0 5 10
field
lake
mickey mouse
soil
grass
green
water
brown
cultivated soil
agriculture
0 5 10 15
river
city
bridge
park
urban
houses
town
water
hosues
parc
0 5 10 15

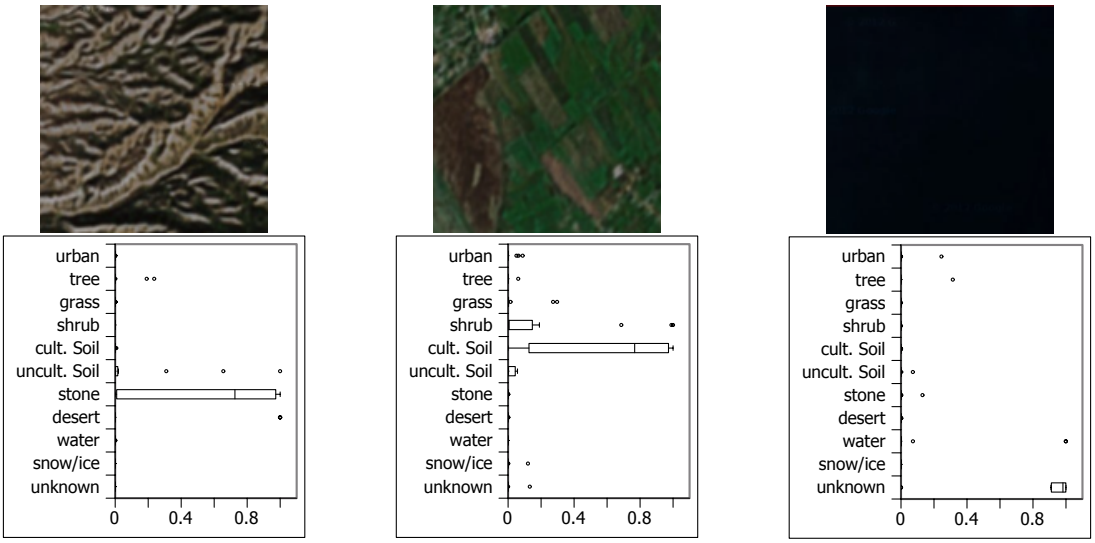

city
urban
rome
houses
trees
town
colosseum
streets
green
street
0 2 4 6 8
dubai
island
water
sea
palm
palm island
beach
harbor
hotel
palmtree
0 5 10
bridge
city
water
sea
harbor
sydney
island
houses
haven
opera
0 2 4 6 8

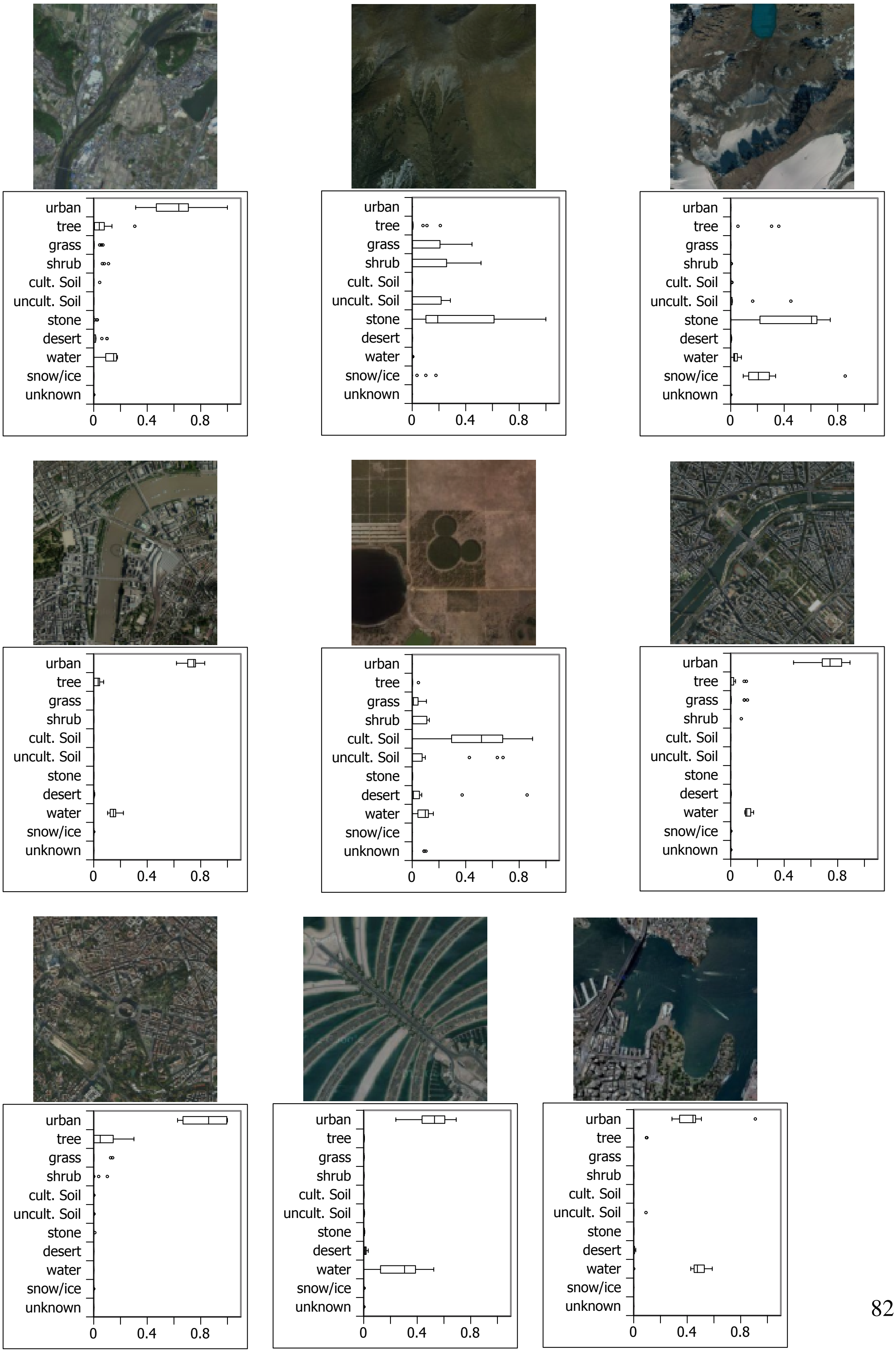

desert
urban
city
pyramids
egypt
gizeh
sahara
soil
field
town
0 2 4 6 8 10
urban
pentagon
city
highway
lake
water
industry
cultivated soil
military
usa
0 5 10 15

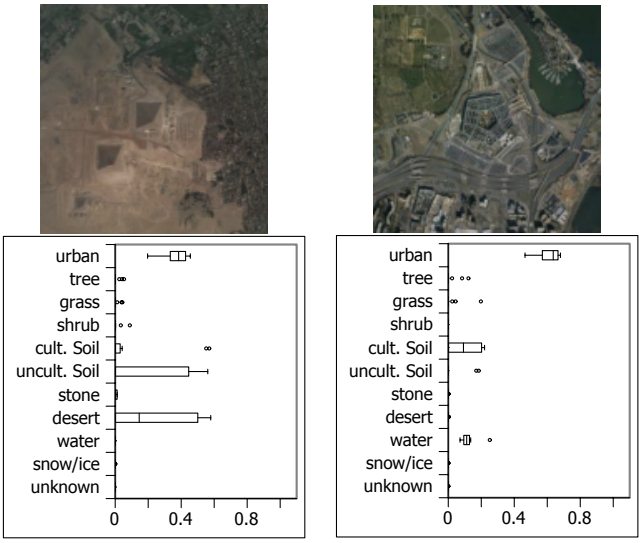
APPENDIX C

All Results of Landcover Drawings

For each landcover map of the test set, a box plot is shown. The lower and upper quartiles form the borders of the box and the median is shown as a horizontal line inside the box. The interquartile range is thus equal to the sidelength of the box. The whiskers at the top and bottom of the box extend to the highest and respectively lowest datum within the one and a half of the interquartile range and outliers are shown as circles. All landcover maps are generated using the Google Maps Javascript API V3. Thus, the copyright holder of the maps is Google inc.


urban
tree
grass
shrub
cult. Soil
uncult. Soil
stone
desert
water
snow/ice
unknown
0
0.4
0.8


urban
tree
grass
shrub
cult. Soil
uncult. Soil
stone
desert
water
snow/ice
unknown
0
0.4
0.8


urban
tree
grass
shrub
cult. Soil
uncult. Soil
stone
desert
water
snow/ice
unknown
0
0.4
0.8
urban
tree
grass
shrub
cult. Soil
uncult. Soil
stone
desert
water
snow/ice
unknown
0
0.4
0.8

Bibliography

[1] European Space Agency. GlobCover. URL https://earth.esa.int/web/guest/missions/esa-operational-eo-missions/envisat.

[2] Institute of Computer Graphics and Algorithms. Landspotting - Creating Games for Improving Land Cover. URL http://www.cg.tuwien.ac.at/research/projects/Landspotting/.

[3] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. SETI@home: an experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, November 2002.

[4] Gary Anthes. HTML5 leads a web revolution. *Communications of the ACM*, 55(7):16, July 2012.

[5] Shyam Boriah, Vipin Kumar, Michael Steinbach, Christopher Potter, and Steven Klooster. Land cover change detection. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '08*, page 857, New York, New York, USA, 2008. ACM Press.

[6] Seth Cooper, David Baker, Zoran Popović, Adrien Treuille, Janos Barbero, Andrew Leaver-Fay, Kathleen Tuite, Firas Khatib, Alex Cho Snyder, Michael Beenen, and David Salesin. The challenge of designing scientific discovery games. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games - FDG '10*, pages 40–47. ACM Press, 2010.

[7] Ryan Dahl. node.js - JSConf, 2009. URL http://s3.amazonaws.com/four.livejournal/20091117/jsconf.pdf.

[8] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. Image retrieval. *ACM Computing Surveys*, 40(2):1–60, April 2008.

[9] Mathias Eitz, James Hays, and Marc Alexa. How Do Humans Sketch Objects? *ACM Trans. Graph. (Proc. SIGGRAPH)*, 31:44:1 – 44:10, 2012.

[10] Express. Express - node.js web application framework. 2012.

[11] Forbes. Draw Something Loses 5M Users a Month After Zynga Purchase, 2012. URL http://www.forbes.com/sites/insertcoin/2012/05/04/draw-something-loses-5m-users-a-month-after-zynga-purchase/.

[12] Steffen Fritz, Ian McCallum, Christian Schill, Christoph Perger, Roland Grillmayer, Frédéric Achard, Florian Kraxner, and Michael Obersteiner. Geo-Wiki.Org: The Use of Crowdsourcing to Improve Global Land Cover. In *Remote Sensing*, volume 1, pages 345–354, August 2009.

[13] Github. Express on GitHub, 2012. URL https://github.com/visionmedia/express.

[14] Github. Socket.IO on GitHub, 2012. URL https://github.com/LearnBoost/socket.io.

[15] GEM GLC2000. http://bioval.jrc.ec.europa.eu/products/glc2000/glc2000.php.

[16] Michael F. Goodchild and J. Alan Glennon. Crowdsourcing geographic information for disaster response: a research frontier. *International Journal of Digital Earth*, 3(3):231–241, September 2010.

[17] Chien-ju Ho, Tao-hsuan Chang, Jong-chuan Lee, Jane Yung-jen Hsu, and Kuan-ta Chen. KissKissBan. In *Proceedings of the ACM SIGKDD Workshop on Human Computation - HCOMP '09*, page 11, New York, New York, USA, 2009. ACM Press.

[18] IDC. Android and iOS Surge to New Smartphone OS Record in Second Quarter, According to IDC, 2012. URL http://www.idc.com/getdoc.jsp?containerId=prUS23638712.

[19] Google Inc. Google Developer - Chrome V8 Design Elements, 2012. URL https://developers.google.com/v8/embed#handles.

[20] Google Inc. Google Developers Embedder's Guide, 2012. URL https://developers.google.com/v8/embed#handles.

[21] Google Inc. Google V8 Engine, 2012. URL https://developers.google.com/v8/intro.

[22] Joyent Inc. Node.js, 2012. URL http://nodejs.org.

[23] Ada P. Kahn and Jan Fawcett. The Encyclopedia of Mental Health - Third Edition. In *Library of Health and Living Set, 53-Titles: 54-Volumes*, page 112. 2007.

[24] Firas Khatib, Frank DiMaio, Seth Cooper, Maciej Kazmierczyk, Miroslaw Gilski, Szymon Krzywda, Helena Zabranska, Iva Pichova, James Thompson, Zoran Popović, Mariusz Jaskolski, and David Baker. Crystal structure of a monomeric retroviral protease solved by protein folding game players. *Nature structural & molecular biology*, 18(10):1175–7, October 2011.

[25] Aniket Kittur, Ed H. Chi, and Bongwon Suh. Crowdsourcing user studies with Mechanical Turk. *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, page 453, 2008.

[26] Zhiwu Lu and Horace H. S. Ip. Automatic Image Annotation Based on Generalized Relevance Models. *Journal of Signal Processing Systems*, 65(1):23–33, October 2010.

[27] Donn Morrison, Stéphane Marchand-Maillet, and Éric Bruno. TagCaptcha. In *Proceedings of the international conference on Multimedia - MM '10*, page 1557, New York, New York, USA, 2010. ACM Press.

[28] NASA. MODIS. URL `http://modis.gsfc.nasa.gov/`.

[29] Vienna University of Technology. Institute of Computer Graphics and Algorithms. URL `http://www.cg.tuwien.ac.at`.

[30] Ryutarou Ohbuchi and Shun Kawamura. Shape-Based Autotagging of 3D Models for Retrieval Autotagging 3D Models via Multiple Manifold Ranking. *SAMT '09 Proceedings of the 4th International Conference on Semantic and Digital Media Technologies: Semantic Multimedi*, pages 137–148, 2009.

[31] Janne Paavilainen. Critical review on video game evaluation heuristics. In *Proceedings of the International Academic Conference on the Future of Game Design and Technology - Futureplay '10*, page 56, New York, New York, USA, 2010. ACM Press.

[32] Zenonas Theodosiou and Nicolas Tsapatsoulis. Crowdsourcing annotation: Modelling keywords using low level features. In *2011 IEEE 5th International Conference on Internet Multimedia Systems Architecture and Application*, pages 1–4, Limassol, Cyprus, December 2011. Ieee.

[33] Stefan Tilkov and Steve Vinoski. Node.js: Using JavaScript to Build High-Performance Network Programs. *IEEE Internet Computing*, 14(6):80–83, November 2010.

[34] Brown University. Computer program can identify rough sketches, 2012. URL `http://news.brown.edu/pressreleases/2012/09/sketches`.

[35] L. von Ahn. Games with a Purpose. *Computer*, 39:92–94, June 2006.

[36] Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the 2004 conference on Human factors in computing systems - CHI '04*, volume 6, pages 319–326, New York, New York, USA, 2004. ACM Press.

[37] Luis von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. reCAPTCHA: human-based character recognition via Web security measures. *Science (New York, N.Y.)*, 321(5895):1465–8, September 2008.

[38] Lei Wang and Latifur Khan. Automatic image annotation and retrieval using weighted feature selection. *Multimedia Tools and Applications*, 29(1):55–71, May 2006.

[39] Xinying Wang. Researches on 3D model automatic annotation and retrieval based on semantic correlation. In *2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering*, pages 274–277. IEEE, August 2010.

[40] WebSocket.org. About WebSocket, 2012. URL `http://www.websocket.org/aboutwebsocket.html`.