UNIVERSITY OF STAVANGER, NORWAY

DAT240 Project Report (Group LauBjuTisVezBra): Image Guessing Game

Martin S. Lauritsen, Alexandre K. Bjugan, Joachim Tisløv,
Kristian Vezina, Ole C. Bråtveit

December 1, 2023

Abstract

In this project, we developed an image guessing game with the objective of engaging two players as a team. The first player assumes the role of the guesser, tasked with identifying the image within the game. The second player acts as the oracle, gradually revealing the image to the guesser piece by piece. In the single-player mode, an artificial intelligence takes on the role of the oracle.

For multiplayer, with multiple guessers collaborating with a single oracle. Additionally, a free-for-all mode allows individual guessers to compete against each other.

CONTENTS

1 Introduction

The inception of this project was fueled by the ambitious goal of crafting an enthralling image guessing game that not only captivates individual players but fosters teamwork and collaboration among participants. In the intricate landscape of our image guessing game, players are invited to embark on a journey where shared insights and collective efforts become the keys to unraveling the visual mysteries before them. Join us as we delve into the immersive world of collaborative gameplay, where every guess, every slice, and every strategic move brings us closer to the ultimate goal of triumph and camaraderie. Welcome to a gaming experience that goes beyond individual prowess, inviting you to explore the endless possibilities that unfold when minds unite in the pursuit of a shared adventure.

2 Project Progression

During the project we employed various tools such as discord, github and docker. Github served as our platform for filedistribution,task allocation, and task assessments. Discord played a primary role in resolving issues, timemanagment, and facilitating basic communication. Finally, Docker served as a container for our application.

3 Team Planning and Collaboration

Our collaboration was well-organized, featuring weekly group meetings. In our initial meeting, we created relationship diagrams, documented them on GitHub, and strategically divided our tasks into smaller work items. This approach allowed for a more focused and manageable pursuit of our objectives. We established deadlines for each individual task and chose to use branching for committing these work items. This division of work not only heightened individual accountability but also facilitated the seamless integration of our efforts.

The incorporation of weekly group meetings provided a platform to discuss both current and future progress, fostering effective communication and alignment of goals among team members. This structured approach significantly contributed to the success and fluidity of our collaborative efforts.

Unfortunately, due to illness and exam scheduling issues, not all deadlines were met on time. This resulted in some delays in work progress and committing to GitHub.

4 Basic Setup

4.1 Session

The Session context was designed with the intent to prepare and compile important settings and information requested and sent by the lobby.

4.2 Frontend Lobby

We have leveraged backend rendering for the majority of the frontend content served to users by the game's backend. This strategy was adopted to reduce reliance on frontend manipulation using JavaScript. Additionally, this method introduces a modest security layer by withholding the rendering of options for any user in the lobby who is not the hosting user.

Although we opted for this approach, a substantial amount of JavaScript was still necessary to prepare for the implementation of multiplayer, SignalR, and its associated functions.

Given more time, one feature that could have been explored is a visual representation explaining why a session host is prohibited from starting a game. However, we considered this to be a cosmetic issue that wasn't deemed a priority so close to the deadline.

4.3 SignalR

We decided to opt for a straightforward implementation of SignalR with a strongly typed hub.

We developed a mapping service that associates the connectionId provided by the SignalR connection with the corresponding user. This mapping service also establishes relationships between individual users and different groups. In our case, each user is limited to being a part of a single group.

The mapping service provides us with the capability to modify the OnConnectAsync function in SignalR. This modification allows the function to update the relationships between ConnectionId and user, ensuring that each user reconnects appropriately to the groups they belong to.

Initially, when initiating the SignalR implementation, we structured it based on the assumption that there would be consistency between User, ConnectionId, and group memberships right from the start. However, this assumption proved incorrect, leading to significant challenges and struggles throughout the SignalR implementation.

5 THE IMAGE GUESSER GAME

In the development of both single and multiplayer games, extensive testing was conducted on various image algorithms, including array manipulation for image merging and sorting. The game's architecture involved querying the [Database Name] to retrieve the names of randomly selected images within a given context, creating an array for reference.

Algorithem Name
Graph Cut Algorithms [1]
Image Stitching with Homography [4]
Wavelet Transforms [5]

Table 5.1: Algorithems tested for image merging

5.1 SINGLEPLAYER GAME

When initiating the single-player mode, the code dynamically queries the database and generates an array containing names of randomly selected image slices relevant to the context. The Oracle/AI then selects a random image from this array, conveying the chosen answer to the backend. This selected image becomes the initial picture loaded on the client side, establishing a visually engaging starting point.

As the player requests an new slice, the AI algorithm selects random a random slice, cross-referencing them with the existing image slice array in the code. If the chosen slice is not present in the array, it is transmitted to the server side and subsequently added to the array. This iterative process continues until the player successfully guesses the image, exhausts all available slices, or decides to conclude the game.

Upon the occurrence of either a correct answer or quitting the game, the calculated score is transmitted from the client to the server side. This score is then sent to the database context for storage.

The implementation of the single-player game utilizes HTML, JavaScript, and CSS. Frontend components play a pivotal role in managing user interactions before ultimately sending the final score to the server side for further processing. This seamless integration ensures a user-friendly and responsive gaming experience, with the backend system efficiently handling scoring events.

5.2 MULTIPLAYER GAME

In multiplayer mode, another human or team player replaces the AI/Oracle. The assigned oracle uses a unique interface to choose image segments as requested by the player. Clicking

on a segment adds it to the "name array," streamlining the transfer of images between the team player and the oracle.

During image loading, the merger combines the segments into a unified image, injecting it into the HTML structure of the oracle. An optimized process checks for the image in an "imageMerged" folder, utilizing it directly if found. If not, the ImageMerger class retrieves, merges, displays, and adds it to the folder for future use, enhancing runtime efficiency and the gaming experience.

In the multiplayer frontend, a polygonal approach offers dynamic user interactions with image segments, adding complexity and strategic decision-making for an enriched gameplay experience.

During the interaction between the client side and the server side, the multiplayer was rewritten to optimize the code. In this more optimized version of the multiplayer, on the Oracle side (the only side that differs from singleplayer), the Oracle can choose an image piece. In this case, ImageSharp is used to calculate transparent pieces and return them to JavaScript, which checks all image pieces to find the piece with non-transparent coordinates that match the mouse coordinates. The main difference between my approach and the first version is the usage of an HTML file called Game.cshtml, whereas the earlier approach had two separate files for multiplayer and multiplayer oracle. This version was fully integrated with SignalR.

6 Advanced Features

6.1 Image Slicing

The group wanted to incorporate the advanced feature to upload their own images to the game, and thereby personalizing their gaming experience. This functionality entails a meticulous two-step process involving the slicing of user-uploaded images and their seamless integration into the game's logic.

Algorithem Name
Seam Carving [2]
Overlapping Block Slicing [6]
Voronoi Diagram [3]

Table 6.1: Other tested slicing algorithms.

To accomplish this, the uploaded images undergo a slicing procedure, resulting in distinct segments that are then individually saved as .png files. These sliced components are subsequently organized and stored in a dedicated folder, with the folder's nomenclature directly corresponding to the name of the uploaded image. This establishes a clear and intuitive connection between the user-uploaded image and the associated gameplay elements.

This sought-after enhancement not only enables players to immerse themselves in a gaming environment featuring images of personal affection but also introduces a layer of customization to the overall gaming experience. By allowing users to add their own image to the game, they get to experience a unique and personalized dimension to the adventure, which then elevates the game to a more individualized and engaging level.

7 Conclusion

7.1 Summary of the Project

This project centers around the development of an engaging Image Guessing Game where players are immersed in a collaborative experience. The essence of the game lies in team dynamics, with players forming alliances either with an artificial intelligence (Oracle) or another human player. The primary objective is for players to deduce the image presented to them, a task made challenging as the Oracle or team player strategically reveals pieces of the image, adding an element of suspense and teamwork to the gaming experience. Through this interactive and collaborative approach, the project seeks to deliver a captivating and dynamic gameplay environment.

7.2 Contributions

Distrobtion of work for each person in the group, was given in table 7.1 .

Name	**Tasks**
Alexander	Advanced Features & Merging
Joachim	Full-Stack
Kristian	Debugging
Martin	Full-Stack & Multiplayer
Ole C.	SinglePlayer & Multiplayer

Table 7.1: The work distrobution

The primary task distribution was as follows: Alexander focused primarily on image slicing and uploading, Joachim's main focus was on the full stack, Kristian concentrated on debugging the code, and Ole Christian focused on both single-player and multiplayer aspects along with Martin. Martin also contributed to the full-stack development in collaboration with Joachim.

7.3 FUTURE DEVELOPMENT

Due to time constraints and the project deadline coinciding with a significant portion of the team's exam period, not all desired goals were achieved. With more time available we would have implemented the already plannned additional advanced features initially envisioned.These features may include enhanced gameplay mechanics, expanded customization options, or the integration of cutting-edge technologies. Additionally, we aim to optimize the existing codebase by refining algorithms, improving system efficiency, and addressing performance areas. This approach aims to create a streamlined and responsive gaming environment, exceeding player expectations. In summary, the future goal would involve integrating advanced features to enhance the gaming experience and optimizing the codebase for a polished and feature-rich version of the Image Guessing Game with an extended timeframe.

REFERENCES

[1] E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. *Computational models of visual processing*, 1991.

[2] S. Avidan and A. Shamir. Seam carving for content-aware image resizing. *ACM Transactions on Graphics (TOG)*, 26(3), 2007.

[3] R. M. R. A. L. C. D. Burrough, Peter A.; McDonnell. Nearest neighbours: Thiessen (dirichlet/voroni) polygons. *Principles of Geographical Information Systems. Oxford University Press*, 2015.

[4] M. U. K. Dmitry Demidov and U. Rahman. Image stitching using homography techniques: Feature extraction and photometric correction methods overview. *ResarchGate*, 2022.

[5] Y. Meyer. Wavelets and operators. *Cambridge University Press*, 1992.

[6] C.-K. C. Pei-Ning Guo and T. Yoshimura. An o-tree representation of non-slicing floorplan and its applications. 1999.