**PROJECT PROPOSAL**

**Snake Hunt: A Competitive Multiplayer Game in Python**

Jaffar Alzeidi

Katrina Janeczko

John Bernardin

Sean Britt

GitHub Repository Link: <https://github.com/cis3296f22/01-snakehunt>

GitHub Project Board Link: <https://github.com/orgs/cis3296f22/projects/97/views/1>

Table of Contents

[Project Proposal 3](#_Toc98789868)

[Project Abstract 3](#_Toc98789869)

[High Level Requirement 3](#_Toc98789870)

[Conceptual Design 3](#_Toc98789871)

[Proof of Concept 3](#_Toc98789872)

[Background 3](#_Toc98789873)

[Required Resources 3](#_Toc98789874)

[Projet Design 4](#_Toc98789875)

[Vision 4](#_Toc98789876)

[Persona Jack, a primary school teacher 4](#_Toc98789877)

[Persona Emma, a history teacher 4](#_Toc98789878)

[Class Diagram 5](#_Toc98789879)

[Project Progress 6](#_Toc98789880)

[Week 2 Progress 6](#_Toc98789881)

## Project Proposal

### Project Abstract

### This document proposes a fun new competitive multiplayer 2D game called Snake Hunt, inspired by multiplayer web game Agar.io and the classic snake game. Users can log into an online server, enter a username, and play against other users. They also have an option to play in single player mode. Each user takes the form of a snake. The user’s goal is to consume food pellets and other snakes in order to grow in length. Once hit by another snake, the user’s snake dies and respawns. Throughout gameplay, their score (length of their snake) is updated on a leaderboard.

### High Level Requirement

*Sample mockup of what gameplay might look like to the user.*

Users play as a snake collecting colorful pellets, navigating throughout the game board using the arrow keys.

*A picture containing light

Description automatically generated*

### Conceptual Design

A player starts the game as a client connecting to the game server. The player begins the game by entering a username (to be used for the leaderboard and for player identification). Upon entering the game, they are assigned a snake of length 1 block and a direction. They begin moving, and cannot stop moving. The player’s snake is always positioned at the center of the screen (POV perspective) and the background changes as the player moves around using the up, down, left and right arrow keys. The borders wrap around such that going off screen to the right results in appearing from the left.

The player begins with a snake that is 1x1 pixel in size and is safe from being consumed (a grace period) until having grown at least to a size of 1x2. To add length to their snake, they must consume food pellets. The snake turns the same color as the pellet it consumes. With each loop of the game, it is checked whether the player’s snake’s head is touching a food pellet in which case, the snake feeds on the pellet.

As a snake grows, every 50th body segment will be labeled as a weak-point by coloring it dark gray.  If a second snake eats the weak point of the first snake, the first snake loses all body segments between the weak point and the tail piece, and the body segment just above the weak point becomes the first snake’s new tail piece. The remains of the removed portion of the first snake are now pellets that can be eaten by any snake on the field.  If the second snake collides with any part of the first snake that isn’t a weak point, it respawns. If both snakes run into each other head on, they both respawn. In both cases, the snakes turn into pellets. When a snake dies, its score (length) is added to a leaderboard which is displayed to the user.

The gameplay is continuous and only ends when the Quit button is pressed, in which the player sees the all-time leaderboard and is then disconnected from and ejected from the game. The option to play again appears in the form of a clickable button.

The hardware requirement to play this game is any computer with a working keyboard, screen and mouse. Speakers for audio functionality is preferred but not required. To build the Snake Hunt Multiplayer Game project, the teammates will need a working computer and GitHub account. The necessary programming language (Python) and libraries are compatible with Windows XP or newer, Linux/UNIX, and macOS. To work on this project, teammates will need to [download Python](https://www.python.org/downloads/) (preferably version 3.7 or newer) and [install Pygame](https://www.pygame.org/wiki/GettingStarted).

To create simple graphics for the game, the [Pygame](https://www.pygame.org/docs/) library will be used. According to the creators, Pygame is a “free and open-source cross-platform library for the development of multimedia applications like video games using Python. It uses the [Simple DirectMedia Layer library](https://www.libsdl.org/) and several other popular libraries to abstract the most common functions”.

Snake Hunt will be a multiplayer game. The goal for a complete and finished project will be hosting the game a webserver for anyone to connect, but this is outside the scope of the project for this class. Until then, the game will run on a local network, such as Temple’s network, meaning no one outside the network will be able to connect. To do this, a server is run to which multiple clients can connect. One computer will need to run the server code and any client will need to connect to the appropriate IP address and port. To implement this, we will be using Python’s [socket module](https://docs.python.org/3/library/socket.html), which provides an interface to the [Berkeley sockets API](https://en.wikipedia.org/wiki/Berkeley_sockets). Please reference the Proof of Concept section for an explanation of how this will work with some example code.

Some other built-in Python modules to be used are random, time, and math.

### Proof of Concept

There is a [public git repository](https://github.com/katrinajaneczko/snake-game) created by a team member that provides the original proof of concept (a separate repository from the current project repository). To create the code for these files, Python’s [socket module](https://docs.python.org/3/library/socket.html) documentation and  a YouTube tutorial by the channel [Tech With Tim](https://www.youtube.com/watch?v=F257x_E6H4k) were referenced. Running this code provides a proof of concept for how Python, Pygame, and sockets will all be able to work together in the creation of a more complicated multiplayer game. It works on Windows and Mac operating systems. The repository must first be cloned. Please see the README file in this repository for further information on requirements on how to run the code. Running it demonstrates a simple version of a single-player snake game as well as a simple client-server prototype of two clients with red and green blocks and sending of positions. It demonstrates simple game logic and classes that will be built upon.

### Background

A simpler version of the original [Agar.io game](https://agar.io/#ffa) was remade [on GitHub](https://github.com/Viliami/agario/blob/master/agar.py). Some of this code can be referenced or reused, especially the way in which the classes are laid out. Many tutorials and versions of the [snake game](https://www.edureka.co/blog/snake-game-with-pygame/) exist online in Python using Pygame and will be referenced for game logic. This game will be very similar to Agar.io and [Slither.io](http://slither.io/) in that it is multiplayer and competitive and consists of players trying to eat one another. It will be different because it will use snakes that grow in length instead of circles that grow in diameter. Also, whenever the player’s snake consumes a food pellet or other snake, it will turn the color of the object consumed.

### Required Resources

Necessary background information includes Python and Pygame as well as resources and information on using Python’s socket library to allow clients to connect to a server. I anticipate race conditions coming up and so we will need to learn more about this as well.

It is also worth discussing here the relevance of this project to the Software Design class. In class, we have discussed multiple topics such as teamwork, version control, testing, and UML diagrams. This project is linked to the educational goals of this class in many ways. It will result in students gaining experience with working in small teams to build a larger project composed of many parts. We will be using Git as version control, so students will become proficient in managing and keep track of modifications to the code using GitHub branching, merging and commit history. It will also give students practice in testing since many tests are going to need to be documented to ensure the code is working in all cases, and TDD may be used for this. Lastly, UML diagrams are a way for students to communicate ideas and the way that all the project pieces will work together, and so we will gain practice in constructing UML class and sequence diagrams.

## Project Design

### Vision

For casual and serious gamers who love to play retro games updated to modern standards, Snake Hunt is a game based on the classic Snake-genre that allows multiple players to play against each other online. Unlike other games in the Snake-genre, Snake Hunt is free-to-play and open-source, allowing anyone with an internet connection to start playing and adding modifications as they see fit.

### Persona Clayton, an esports enthusiast

Clayton is a 19 year old college student from Swampscott Massachusetts. He currently goes to Arcadia University and is on their League of Legends and Hearthstone teams. He is a competitive gamer who tries his absolute best to win against other players with his high end pc. The Snake Hunt game is perfect for Clayton because he can strive to be at the top of the global leaderboards by defeating all other opponents.

### Persona Thomas, a grandfather

Thomas is 52 years old and has just been gifted a computer by his son-in-law, Jim, who has shown Thomas how to play a new game called Snake Hunt. Nostalgic for the snake game he grew up with, he enjoys the new multiplayer version which still gives a nod to the retro feel and that he can play with his son-in-law. Thomas looks forward to playing with his future grandchildren one day.

**Persona Rachel, a casual low-stakes gamer**

Rachel is a 15 year old who usually plays single-player games. She has tried out a few multiplayer games but most of them were too competitive and difficult for her taste. She is looking for a multiplayer game that has minimal competition. Snake Hunt is perfect for that, because she can choose to play passively by avoiding other snakes and looking around for pellets.

**Persona Jimmy, a young procrastinator**

Jimmy is a 12 year old from Seattle Washington who is just getting into video games. He plays on a fairly out of date laptop that his mom used to have. He has found a new online multiplayer computer game from his new friends at school. When he needs a break from homework or just really wants to tune out of what his teachers are saying during class, he can relax and zone out by playing this easy-to-learn, beginner-friendly game. The continuous gameplay makes this game a fun way to procrastinate with friends.

### Class Diagram

*Diagram, schematic

Description automatically generated*

Fourteen total classes across multiple files come together to make this complete class UML diagram. Server aggregates multiple Players, 1 Game, and 1 RandomPellets, while Player aggregates 1 Snake, which aggregates multiple BodyParts. Player uses Message. RandomPellets includes Pellet. Game includes 1 Camera and 1 PauseMenu. Client uses CellData, LeaderBoardEntry, and GameData.

### Sequence Diagrams

*Diagram

Description automatically generated***UML Sequence Diagram for Single-Player Game Loop**

During the game loop, the pause menu is instantiated. All other game processes are halted until the user inputs a user name and presses OK. Then, user input from the arrow or WASD keys will cause the snake to move, which then causes each particular body part to move. Then if the user maneuvers the snake into colliding with a pellet, the snake's size increases by one and a body part is appended to the snake, then the pellet's position is reset. Additionally, if the user maneuvers the snake into colliding with itself, it will have its size reset to 1. Finally, if the user clicks the x button on the game window, the loop terminates as will the program.

### UML Sequence Diagram for Name Selection

*Diagram

Description automatically generated*

Server starts via main(), then listens for connections through the Socket. Upon accepting, a Client connects. The method input\_addr() is called on itself, then connect() is called to establish a connection to the Socket. Client activates PuaseMenu and Server calls receive\_name() on itself to get the user’s name. PauseMenu calls send\_data(name), sending the name data to Comm, and Server calls receive\_data() then send\_data(TRUE) to Comm (or FALSE if the name is not valid). If true, the receive\_name() function returns and receive\_data() is called by PauseMenu of Comm, and once it returns, PauseMenu() returns to Client.main(). A name has been chosen and set for the Client.

### Automated Test Results

A picture containing text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

## Project Progress

### Week 2 Progress

**Sprint Goal:** The goal of this sprint is to expand quality-of-life and user experience features of the single-player snake hunt game.

**Backlog Features**

* POV Camera
* Pellet Consumption
* Leaderboard
* Changing Background/Play Area
* Multiplayer

|  |  |  |
| --- | --- | --- |
| Tasks in Sprint | Task Status at end of Sprint | Assigned To |
| Increase play area | Completed | Sean |
| Create POV camera | Completed | Sean |
| Complete week 2 doc | Completed | Sean |
| Spawn multiple pellets | Completed | John |
| Leaderboard creation | Partially completed | Jaffar |
| Create single-player UML class diagram | Completed | Jaffar |
| Update single-player UML class diagram | Completed | John |
| Create executables and issue 2nd release | Completed | Jaffar |
| Name selection | Partially completed | Jaffar |
| Client-server 2-player squares prototype | Completed | Katrina |
| Multiplayer (>2) functionality with squares | Completed | Katrina |
| Client-server 2-player with snakes | Not started | Katrina |
| Edit week 2 README | Completed | Katrina |
| Refactor single-player snake code | Not started | Jaffar |

### Week 3 Progress

**Sprint Goal:** The goal of this sprint is to explore and find a way to integrate the client-server prototype with the existing single-player snake game code.

**Backlog Features**

* Shed skin
* Multiplayer

|  |  |  |  |
| --- | --- | --- | --- |
| Tasks in Sprint | Size | Task Status  at end of Sprint | Assigned To |
| Refactor snake game code | 3 | Completed | Jaffar |
| Exploratory activity 1 with client-server & snakes | 5 | Completed | Katrina |
| Exploratory activity 2 with client-server & snakes | 3 | Completed | Sean |
| Exploratory activity 3 with client-server & snakes | 8 | Completed | Jaffar |
| Exploratory activity 4 with client-server & snakes | 5 | Partially completed | John |
| Create 3rd release | 1 | Completed | Jaffar |
| Create sequence diagram #2 with single-player snake | 2 | Partially completed | John |
| Create sequence diagram #1 with multiplayer squares | 2 | Completed | Katrina |
| Update design doc for week 3 | 2 | Completed | Katrina |

|  |  |
| --- | --- |
| **Estimated Velocity**  (At beginning of Sprint) | 34 |
| **Calculated Velocity**  (At end of Sprint) | 31 |

### Week 4 Progress

**Sprint Goal:** The goal of this sprint was to integrate code from the single-player snake game prototype and the multiplayer client-server prototype to create a working multiplayer game with full updated functionality.

**Backlog Features**

* User interface/buttons

|  |  |  |  |
| --- | --- | --- | --- |
| Tasks in Sprint | Size | Task Status  at end of Sprint | Assigned To |
| Combine Client/Server with Main Branch | 5 | Completed | Jaffar |
| Shed skin | 3 | Completed | Sean |
| UML Class diagram | 4 | Completed | Katrina |
| Update UML sequence diagram #1 | 2 | Completed | John |
| Update UML sequence diagram #2 | 3 | Completed | Jaffar |
| Update UML sequence diagrams with descriptions | 1 | Completed | Katrina |
| Create Mac executable | 1 | Completed | Sean |
| Create Windows executable | 1 | Completed | Jaffar |
| Publish 4th release (V1.2.1) | 1 | Completed | Sean |
| PyDoc API documentation | 2 | Partially Completed | John |
| Write unit tests for single-player | 3 | Completed | John |
| Write unit tests for multiplayer | 5 | Not started | None |
| Export coverage report | 1 | Completed | Katrina |
| Export unit test results | 1 | Partially completed | John |
| Update design doc for week 4 | 2 | Completed | Katrina |
| Calculate velocities | 1 | Completed | Katrina |

|  |  |
| --- | --- |
| **Estimated Velocity**  (At beginning of Sprint) | 33.5  (31+36)/2 |
| **Calculated Velocity**  (At end of Sprint) | 30.5  (31+30)/2 |