Doki Doki Fight Club

COMP8037 Major Project Proposal

Alexis C. Mendiola – A01089168

11-19-2022

Table of Contents

1. Student Background

Alexis C. Mendiola (hereafter referred to as “the developer”) is currently a student of the Bachelor of Technology (BTech), Games Development Option at the British Columbia Institute of Technology (BCIT). She has always loved video games; this love drove her to pursue a career in game development. She entered BCIT’s Computer Systems Technology (CST) Diploma in January 2019 and graduated from the Technical Programming Option With Distinction, after which she immediately entered BTech. At BCIT, the developer became proficient in Java, C#, and Python. Additionally, she has experience with JavaScript and other web development tools (e.g. Node.js).

1.1. Education

Bachelor of Technology, Games Development Option *2021–Present*
British Columbia Institute of Technology

- COMP8051 Advanced Games Architecture
- COMP8082 Project Management
- COMP7904 Storytelling Techniques in Games Development
- COMP7051 Introduction to Computer Games Development
- COMP7903 Game Design Fundamentals

Computer Systems Technology (CST), Technical Programming Option *2019–2021*
British Columbia Institute of Technology

- COMP4537 Internet Software Architecture
- COMP4952 Human-Computer Interaction for Application Development
- COMP4956 Software Engineering
- COMP2522 & COMP3522 Object Oriented Programming 1 & 2

1.2. Work Experience

“BCIT Sessions” – Industry Sponsored Student Project *Sep. 2020–Dec. 2020*
Christopher Thompson, Instructor at BCIT

- Proof of Concept
- A web application that allows students to register themselves in a real-time queue to have online meetings with instructors
- Created with Node.js, MySQL, ReactJS

2. Project Description

Doki Doki Fight Club (DDFC) is a three-dimensional (3D) online game in which players are against one another in a 1v1 fight to the death. DDFC will utilize biometric[1] devices (i.e. smartwatches) to monitor player heart rates (HRs). The HRs will be translated into heartbeat sound effects that the enemy can hear if they are in proximity. Players who can manage their feelings of fear and keep their HR low will gain an advantage over their opponents by being less detectable. DDFC will feature low visibility (e.g. dim lighting, muted colour palette, et cetera), so players rely more on audio cues. Other games that currently utilize HR monitors are *Nevermind* and *Bring to Light*. Both are single-player horror games, and the devices required are worth several hundred dollars. DDFC aims to solve the problem of inaccessibility by utilizing smartwatches (instead of Intel RealSense or Virtual Reality (VR) headsets). DDFC's online aspect will provide a novel experience for players; players will know whether or not they strike fear in the heart of their opponent. This project will provide technical challenges because the developer has yet to develop a networked multiplayer game or utilize biometrics. Initial research suggests that DDFC will be the only game that uses heartbeat audio from another player as part of the gameplay mechanics; existing games either do not use the heartbeat audio at all or to a minimal degree.

2.1. Game Instructions & Rules

2.1.1. Controls

Keyboard & Mouse

- *WASD* - **Move**
  - *W* - Forward
  - *A* - Left
  - *S* - Backward
  - *D* - Right
- *C* - **Crouch**
- *Shift* - **Change Speed**
  - Default is Walking
  - *Shift* (Hold or Toggle) to Sprint
  - Players can change the Default Speed in the Settings
- *Space* - **Jump**
- *Mouse* - **Look Around**
- *Left Mouse Button (LMB)* - **Quick Attack**
- *Right Mouse Button* - **Heavy Attack**

[1] Biometrics are measurements or calculations of human characteristics (e.g., reading a heart rate).

2.1.2. Goals

Player goals are divided into three categories: Macro Level, Micro Level, and Short-Term. The Macro Level goal is the game's final objective (e.g., win the game, complete all puzzles, etc.). Micro Level goals are the player's moment-to-moment objectives, such as dodging an attack, jumping onto a platform, collecting a power-up, and so on. A Short-Term goal is somewhere between a Macro and Micro Level goal, and they are essentially a collection of Mirco Level goals that help them attain a prerequisite to achieve the Macro goal. An example of a Short-Term goal is beating a single level in a game with ten levels.

Below is a description of a few of the players' goals in DDFC:

- **Macro:** Defeat the enemy player
- **Short-Term**: Find the enemy player
- **Micro:** Dodge attacks, listen for the enemy's heartbeat, maintain a low heart rate

2.1.3. Win/Lose Conditions

Win Conditions

- **Match**
    - Win two of the three Rounds that must be played in a single Match.
- **Round**
    - Within the time limit, reduce the enemy player's health to zero while maintaining at least one health point (i.e. defeat the enemy before the enemy can defeat you).
    - If both players are alive when the time limit expires, the player must have one or more health point(s) than the enemy.

Lose Conditions

- **Match**
    - Lose two of the three Rounds that must be played in a single Match.
    - The player disconnects from the match while it is in session.
- **Round**
    - Fail to reduce the enemy player's health to zero once the time expires.
    - If both players are alive when the time limit expires, the player has fewer health points than the enemy.

2.2. Game Design

DDFC is a 1v1 death match in which players must master their sense of fear to master the game. The calmer a player remains, the quieter their heartbeat will be to enemy ears. Players must strategize by being aware of how much noise they are making, traversing the map with intentionality (i.e., choosing to walk in certain situations or running in others), and carefully considering how much damage each of their melee attacks does. DDFC's features will be outlined in the following sections.

This section describes DDFC's design using the MDA Framework described in *MDA: A Formal Approach to Game Design and Game Research* by Robin Hunicke, Marc LeBlanc, and Robert Zubek. The MDA Framework is composed of Mechanics, Dynamics, and Aesthetics. Hunicke et al. (2004) define these terms as follows:

> **Mechanics** describes the particular components of the game, at the level of data representation and algorithms.
>
> **Dynamics** describes the run-time behavior of the mechanics acting on player inputs and each others' outputs over time.
>
> **Aesthetics** describes the desirable emotional responses evoked in the player, when she interacts with the game system.

2.2.1. Mechanics

Mechanics can typically be described as an action that is performed by the game's logic (especially in response to player inputs).

- Move
  - Change Speed (Walk/Run)
- Jump
- Crouch
- Attack
- Heart Rate

2.2.2. Dynamics

The Dynamics will describe how DDFC's mechanics can be used by the player to play the game in specific ways (i.e., employ strategies).

- Change speeds to move slower and quieter, or quicker and louder
- Crouch behind objects on the map to avoid detection
- Remain calm to maintain a lower HR and limit the noise made by the player's heartbeat

2.2.3. Aesthetics

DDFC will feature low-poly models (LPM). An LPM "is usually described as a 3D model with a low number of polygons. Polygons are geometrical shapes (often triangles or quadrilaterals) that make up a polygon mesh and are responsible for creating details in a 3D model" (Game-Ace, 2021). The benefit of low-poly models is the abundance of free resources, such as those made by Quaternius (https://quaternius.com/index.html) or those available in the Unity Asset Store (https://assetstore.unity.com/). It will be easier to find external assets with similar styles using LPMs instead of high-poly models.

Figure 1

Note: Image of a low-poly model by Quaternius (https://quaternius.com/packs/ultimatemodularcharacters.html)

DDFC will have a similar visual aesthetic to the Gulag featured in *Call of Duty: Warzone* (see Figure 2 below). The lighting will be limited throughout the map, resulting in a dimly-lit environment. Players need to rely more on their sense of hearing to find the enemy as opposed to relying on visual cues like in other player-versus-player (PVP) games.

Figure 2

Note: Promotional Image of the "Showers" Gulag from *Call of Duty: Mobile* (https://callofduty.fandom.com/wiki/Gulag_Showers)

The colour palette will be primarily made up of muted colours and earth tones, resulting in a grittier aesthetic. The design decision regarding the aesthetic was made when considering player immersion.

Figure 3

Note: Image of example colour palette.

2.2.4. Initial Prototype

Before beginning the project's development stage, a wireframe prototype will be created as a proof of concept and initial design. The prototype will be developed using Figma (https://www.figma.com/prototyping/). Note that the prototype will not be a functioning game, and any designs may change when development begins.

The prototype's features will demonstrate the following:

- UI elements
- Map design
- Gameplay concept image

2.3. Target Audience

DDFC is a tactical fighting game, unlike *Tekken* or *Mortal Kombat*. In *Tekken* and *Mortal Kombat*, players are required to input a combination of attacks and quickly respond to the enemy's actions. In DDFC, Players' health will be limited; therefore, few attacks are required to kill enemies. The gameplay will be slower-paced than traditional fighting games and require more patience. Therefore, DDFC will cater to a mature audience of an older age group. Bartle's Player Taxonomy, DGD1 Play Style, LeBlanc's Taxonomy of Game Pleasures, and the Myers-Briggs Type Indicator (MBTI) will be used when describing the Player Profile.

Bartle's Taxonomy divides players into categories based on their personality and how they like to interact with a game or its other players. The Bartle Player Types are as follows: the Socializer, the Achiever, the Explorer, and the Killer. Killers "get a thrill from gaining points and winning status … They're highly competitive, and winning is what motivates them" (Kumar et al., 2020).

DGD1 refers to the First Demographic Game Design Model created by Chris Bateman and Richard Boon in 2005. DGD1 is "an adaptation of Myers-Briggs typology to games, and thus an investigation of how the patterns within this inventory applied to playing games" (Bateman et

al., 2011). The four play styles found in their research were Conqueror, Manager, Wanderer, and Participant. The Conqueror type "means that the player wants to be the last one standing, to be able to beat the other players" (Thomét, 2012). Bateman and Boon also discuss the difference between a 'casual' player and a 'hardcore' player, where casual players are those who do not devote as much time or money to games. Being a casual player does not mean the player enjoys playing games less than a hardcore player.

LeBlanc's Game Pleasures (also referred to as Aesthetics) describes eight ways players can experience games. Players who prefer a Challenge aesthetic prefer games that, unsurprisingly, make them feel challenged. Fighting games or First-Person Shooters typically aim for a Challenge aesthetic.

The MBTI has sixteen personality types. Personalities are determined by a person's preferences in the following categories (The Myers & Briggs Foundation, n.d.):

- Favourite world
- Information
- Decisions
- Structure

DDFC players primarily have an ISTP or ISTJ personality, which translates to Introvert, Sensing, Thinking, and Perceiving/Judging.

2.3.1. Player Profile

The player profile will be written in the form of a User Persona. "User personas are archetypical users whose goals and characteristics represent the needs of a larger group of users" (Faller & Babich, 2019). Using a User Persona helps generate a better understanding of the target audience.

Age Range	17-40	**Casual vs. Hardcore**	Casual
DGD1 Playstyle	Conqueror	**LeBlanc Aesthetic**	Challenge
Bartle Player Type	Killer	**MBTI**	ISTP/ISTJ

2.3.2. User Persona

David Park is a 23-year-old university student who occasionally wants a break from the seemingly never-ending pile of homework. He doesn't want to fall behind in school, so he likes playing games that are easy to pick up and put down. David is competitive, indicated by his ISTJ personality type and LeBlanc aesthetic. David likes the idea of challenging himself and pitting his game skills against another player because he loves the opportunity to come out victorious. Because David doesn't work while schooling, he doesn't play games that require expensive hardware (e.g. VR headsets). He already owns a Fitbit because of his desire to keep track of his fitness, so a game like DDFC is ideal for him.

3. Problem Statement and Background

Biometric feedback (see Footer 1) is uncommon in video games; measuring a player's HR during gameplay is even more so. As mentioned earlier, two examples of games that monitor a player's HR for use in-game are *Nevermind* and *Bring to Light*. Both games only support single-player gameplay and require expensive hardware, such as clunky chest straps or Intel RealSense devices to respectively detect player HR or emotions (via microexpressions). These devices are very specialized and would not have many other practical uses.

These games have certainly made innovations by utilizing HRs. Still, they have not pushed the letter beyond using the player's heartbeat as a backing soundtrack (*Nevermind*) or introducing a grainy, horror film-like overlay (*Bring to Light*). The game features tied to the player's HR are not very connected to core gameplay mechanics. For example, the HR is used as background audio in *Nevermind*, but the player's HR does not matter. The HR could be 130 Beats Per Minute (BPM) or 80 BPM, and there is no difference in how the game is played. As a result, some players seemed underwhelmed by the inclusion of biometric feedback.

DDFC aims to address these problems by adding significance to the data read from the biometric device and by being more accessible to players. DDFC will have compatibility with Fitbit watches, which are generally less expensive than the devices used in existing biometric games. Additionally, DDFC's gameplay will change depending on how fast or slow a player's heart is beating. This adds a more interesting dynamic to reading a player's HR. More detail regarding DDFC's scope and features can be found in Sections 2, 4, and 6.

In the event that the player's HR data is not successfully retrieved from the Fitbit Web Application Programming Interface (API) or fails to transmit to the enemy player's client, DDFC will continue using the last-used audio levels and playback speed. Because players cannot connect to a game session without their resting HR and current HR being detected, DDFC will always have default data to fall back on. Furthermore, players will be punished with Round Losses if continual device removal is detected by the Fitbit Web API.

4. Scope and Depth

DDFC is a multiplayer game in which players will participate in a 1v1 death match. Therefore, the required systems will include networking, game logic, and persistent data. Later in this section, these requirements will be explained in further detail.

4.1. Planned Features

- Target Only PC Platforms (Windows and macOS)
- A Single Map
- Entity-Component System Architecture (via Unity)
    - Stats
        - Player Health
        - Damage

        - Speed
    - NetworkManager
        - Manages the networking aspects, such as data that must be sent/received by each player's client (e.g., Enemy actions such as movements, attacks, and position)
- Persistent Data
    - Players' statistics
- Melee Combat System
- 3D models
- Consumption of Fitbit Web API
    - Retrieve players' Heart Rate data (i.e., HR in BPM, resting HR)
- Simple Matchmaking

DDFC will feature **a single map** design. Players will have the ability to connect to a game session that holds an instance of this map. Initially, **the network topology** is planned to be either a **Dedicated Game Server (DGS) or Client-Hosted Listen Server**. When the project is closer to its development stages, the developer will do more research on which topology is most suitable. Although, initial research suggests that a DGS may be preferred.

Player HR data will be consumed from the Fitbit Web API and transmitted across the server to the opposing player's client, where it will be converted into a dynamic sound effect. Players' HRs will determine how loud the audio cue is on the enemy's client; a higher HR results in louder audio and a lower HR results in quieter audio. Player combat will also be **restricted to melee (close-range) combat**.

Persistent data will consist only of a **player's statistics, preferences, and a client identification (ID) number.** Stats will include wins, losses, total damage dealt/received, et cetera. The player's preferred settings (such as overall volume and mouse sensitivity) will be stored locally, along with the unique client ID. The server will utilize the client ID to connect the player to game sessions.

Lastly, the developer will **implement simple matchmaking** when connecting players to available game sessions. Factors such as skill will not be factored into the algorithm (see below: Section 4.2.).

4.2. Out of Scope

- Cross-platform compatibility
- Multiple dedicated servers
- Global leaderboards
- Skill-based matchmaking algorithms
- Projectiles/Long-range combat

Due to time constraints, certain features or concerns are out of scope and, therefore, will not be implemented in the final product. First and foremost is cross-platform compatibility. DDFC will

support Windows and macOS; it will not support mobile, console, or Linux platforms. Next, the developer will restrict the chosen network topology to free solutions, potentially limiting the number of (dedicated) servers or the reliability of the connections. The remaining feature limitations are regarding the game logic. As previously mentioned, DDFC will have only one map, which ensures that more focus can go into implementing the network and integrating the systems correctly. Additionally, there will be no database storing global leaderboards. More importantly, DDFC will not feature skill-based matchmaking algorithms when players search for a game session. Lastly, the combat systems will not include projectiles (i.e. long-range combat) to simplify the player input that must be communicated across the network.

4.2.1. Stretch Goals

Stretch goals will be considered for implementation only once all planned features have been implemented (or removed from the scope for good reason). These goals will be used to provide extra challenges and bring more value to the project.

Current Stretch Goal(s)

- Additional map
- Algorithm for selecting which map to load

4.3. Feasibility

After initial research, the available technologies and frameworks support the feasibility of the proposed implementation. There are some minor uncertainties regarding the exact implementation of certain systems. However, overall, DDFC is technically feasible. For example, Unity applications that interface with the Fitbit API already exist. The developer must determine the optimal plugins and frameworks that are needed to ensure the project's success.

5. Test Plan

The developer will utilize several testing procedures to ensure the quality of the work. The following test plan may be subject to change once the project nears the development phase, as different technologies may require alternative testing methods or frameworks.

5.1. Testing Procedures

Manual Testing	Involves manually debugging and compiling code without automation
Integration Testing	Involves testing the links between system modules for successful end-to-end functionality
Playtesting (User Testing)	Involves silently observing players as they play a prototype, followed

	by a survey or interview

As part of the manual testing, the developer will perform regression tests in each Milestone to ensure that the project still meets all functional requirements.

5.2. Test Tools

The following is a preliminary list of tools the developer expects to use during testing. However, it is not an exhaustive list and is subject to change closer to the development phase.

- Automation (e.g. for Unit Testing) will likely use Unity Test Framework or AltUnity Tools
- Postman will be used for testing API calls
- Wireshark will be used for network analysis

5.3. Example Test Cases

The following is an inexhaustive list of potential test cases where the “expected result” is akin to a “pass condition.”

Clients Connecting

Case 1	Client connects to a new game session
Description	The player launches the game and attempts to connect to an existing session.
Expected Result	The endpoint successfully responds to the network request. The player’s unique client ID is registered, then the client loads the game session and spawns a player model.

Player Heart Rate Detection

Case 1	Player’s heart rate is detected
Description	The player’s Fitbit device is already authorized and connected to the game session. The application then requests heart rate data from the Fitbit Web API.
Expected Result	Fitbit Web API successfully returns heart rate data

Case 2	Client receives enemy player’s heart rate data from the server
Description	The player is within audible distance of their enemy. The player’s client calls the server to receive the enemy’s heart rate data, which will then be played as an audio cue.
Expected Result	The data is successfully parsed. An audio cue that corresponds to the detected BPM of the enemy player is played on the client.

5.3.1. Functional Requirements

	Requirement
Build	The application must run on both Windows and macOS platforms
Authentication	The system must allow users to authenticate their Fitbit device by redirecting them to a browser login page
Network	The system must load a map when a session is begun The system must allow two players to connect to a game session The system must detect when a player disconnects from a session before the match ends The system must retrieve the player's heart rate from the Fitbit API The system must transmit player data (Health, Heart Rate, etc.) to the server
Player	The system must move the character model according to user movement inputs
	The system must apply the appropriate amount of damage when the player is attacked or performs an attack

5.4. Example Playtest

The developer has access to approximately ten playtesters (the approximation is due to the availability of resources such as time and sufficient Fitbit devices at the time of testing). Six of the ten playtesters match three or more of the Player Profile components (see Section 2.3.). More information regarding each playtester will be provided in a document compiled after each play session, in addition to the developer's notes and the post-playtest interview results.

The tester will be given a playable prototype. Before starting the play session, the developer will provide the game instructions and rules to the tester (see Section 2.1.). The tester will be asked to play the game while speaking their thoughts aloud. During this play session, the developer will avoid interrupting unless necessary to ensure that the tester is as uninfluenced as possible. The tester's observed thoughts, actions, body language, and general emotional state will be noted down during the session. These notes will be used to determine if certain areas of the game were interesting, boring, frustrating, et cetera. Once the play session has concluded, the tester will be interviewed or given a survey to gather additional feedback.

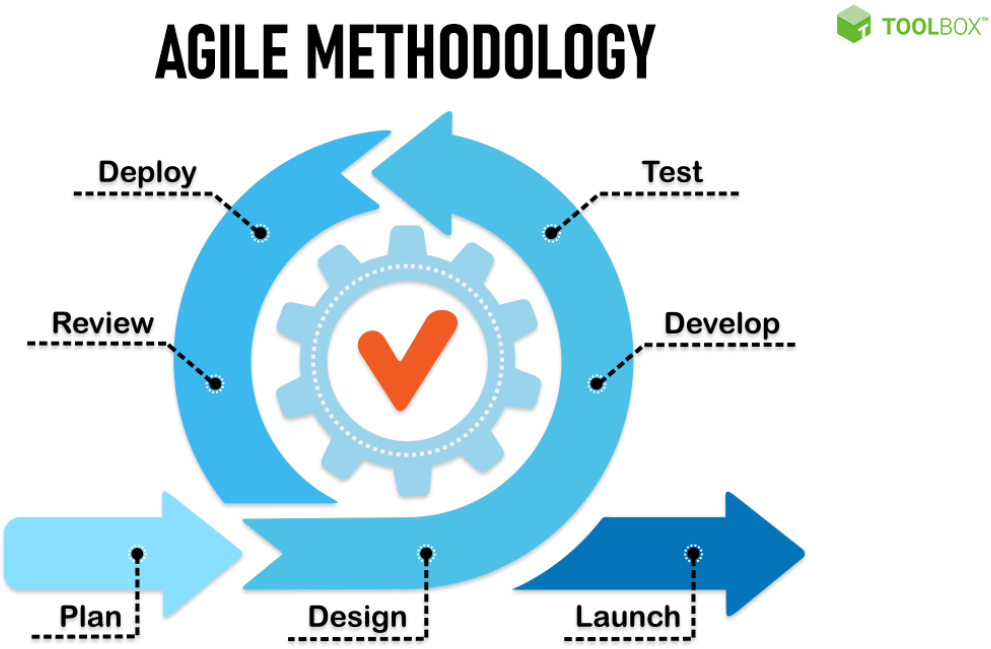
5.4.1. Post-Playtest Interview Questions

- What did you think of the time-to-kill (TTK)?
- How were the audio levels?
- On a scale of 1 to 10, how would you rate the combat difficulty?
- What did you think about the map's layout?
- What aspect of the game stood out the most (positively or negatively)?

6. Methodology

Agile methodology will be utilized to iteratively approach the project's software development cycle. Using Agile, the developer will break the workflow into biweekly Sprints in which she will iteratively develop DDFC. Generally, Agile development follows a cycle of planning, designing, developing, testing, deploying, then reviewing (as seen in Figure 4 below).

Figure 4



Note. Diagram of Agile methodology and software development cycle.

DDFC's development will also follow this iterative workflow. The developer will go through all of these phases within each Sprint. By the end of each Sprint, the developer expects to have implemented all systems required for a functional prototype with end-to-end connectivity.

6.1. Approach

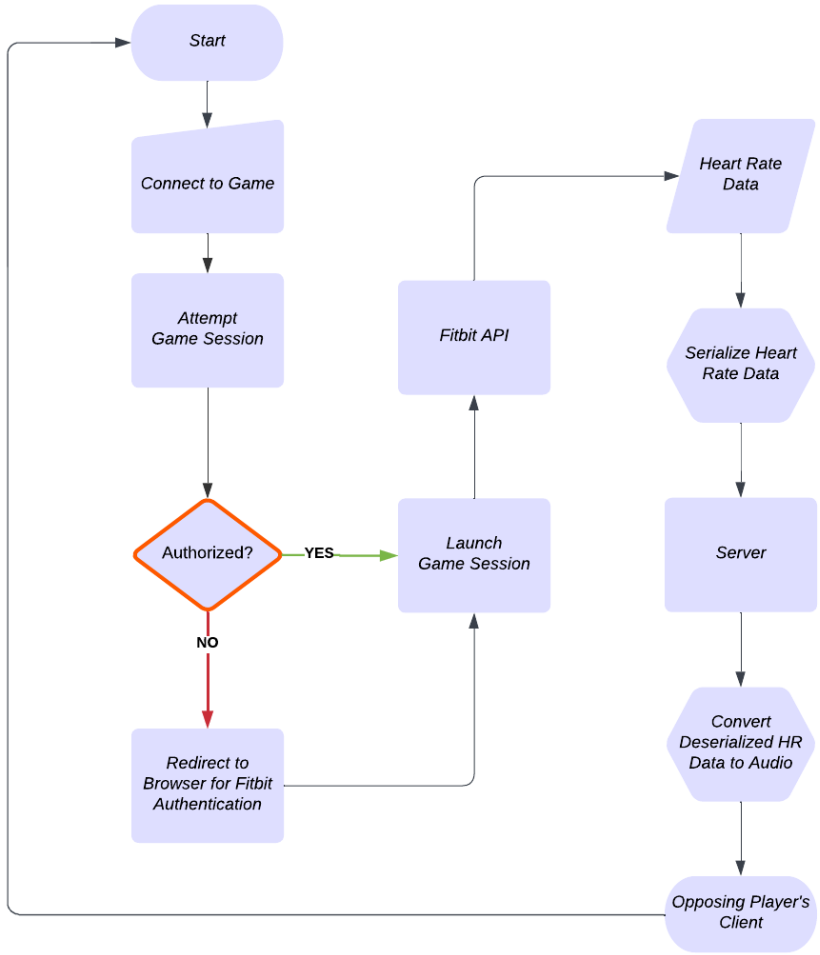
The developer will utilize Agile and Scrum methodologies. Using Agile, the developer will iteratively produce a working prototype on a biweekly Sprint schedule. The Sprint will begin with a planning phase, in which the developer will move tasks from the Product Backlog (a list of all tasks required for the project's completion) to the Sprint Backlog (a list of tasks to be completed during the sprint). The developer will use Trello for task management. Once the developer selects tasks to complete during the Sprint, she will design, develop, and test all code. The developer will repeat this process for each Sprint until the project's completion.

6.2. Technologies

- Unity Game Engine (version 2021.3.11f LTS)
- C# language (version 8.0) for writing game logic
- Mirror Networking will be used for multiplayer networking logic
- GitHub Repository will be used for software version control
- Trello will be used for Sprint planning, and task management
- Fitbit Web API will retrieve players' heart rate data and communicate it to the game
- Google Docs will be used for documentation

7. System/Software Architecture Diagram

Figure 5

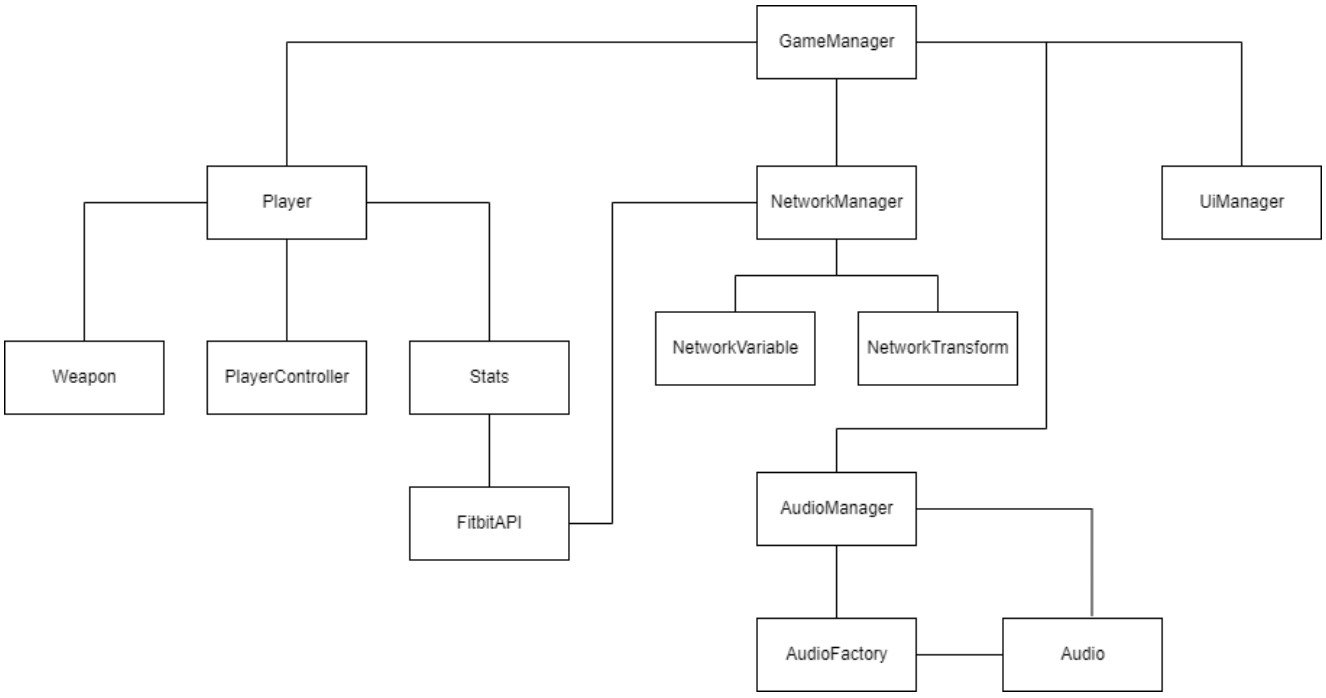


Note. Flowchart Diagram that describes the flow of a player starting the game and how the systems interact.

As depicted in Figure 5, when a player launches the game, the first system they will encounter is the authentication for their Fitbit account. This process will be initiated by making an authorization call to the Fitbit Web API, after which the game will redirect the player to a browser. Once the user provides their authentication, they will be redirected back to the game with a client secret and token. These tokens will be used when the client makes future calls to the Fitbit Web API for the player's HR data. The HR data will be serialized and sent across the multiplayer network to later be converted into a soundbite (or audio cue) that corresponds to the HR. The audio cue (the sound of the player's heartbeat) plays on the opposing player's client. This process, excluding the initial Fitbit account authorization, will be repeated throughout the

two players' match. Their HR data will continually be retrieved and manipulated into audio cues, then sent back and forth across the network.

Figure 5



Note. UML Class Diagram, depicting the projected system architecture.

7.1. Heart Rate Data Conversion to Audio Levels (Algorithm)

Players' resting HRs and last recorded HRs (in BPM) will be retrieved using the Fitbit API. Fitbit calculates the resting HR by using HR data gathered throughout the entire day. Players' resting HRs will be recorded when they first connect to a game session; this recorded value will be used as a baseline and be considered what is "normal" for the player. The audio levels of the heartbeat SFX that is audible to enemies will be calculated using the difference between their most recently recorded HR and the initially recorded resting HR.

Every five seconds, each player's client will call the Fitbit API to update the player's current HR. When the player's HR is retrieved from the Fitbit API during an active game session, the HR will be compared against the resting HR. The audio level of the heartbeat sound effect will be increased or decreased according to the difference between the player's resting HR and their updated (current) HR.

Below is pseudocode demonstrating a potential implementation of the algorithm:

```
void ConvertAudioLevels()
{
        restingHR = GetRestingHR()
        currentHR = GetCurrentHR()
```

```
        // Calculate in/decrease of currentHR compared to initial restingHR
        hrPctDiff = ((restingHR - currentHR) / |restingHR|) * 100

        // Get difference between percent increase of HR and audio volume percent
        volPctDiff = (hrPctDiff - CurrentAudioLevelPct) / CurrentAudioLevelPct * 100

        AdjustAudioVolumeLevel(volPctDiff)
        AdjustAudioPlaybackSpeed(volPctDiff)
}
```

8. Innovation

Biometrics is not new to video games; neither is the use of HR sensors. What makes DDFC innovative is how the game will incorporate the sound of players' heartbeats. Other games use biometric feedback to play the sound of the player's heartbeat in the background or to add a visual effect on the screen. DDFC will go one step further by incorporating players' heartbeats in core gameplay mechanics. DDFC will utilize the Fitbit API to gather HR data from each player's device and transmit it across a multiplayer network. DDFC will convert the HR data into an audio cue on the opposing player's client. The ability to hear the *enemy player's* heartbeat will be unique to DDFC. Additionally, all games that use HR sensors are single-player, but DDFC will be multiplayer.

9. Complexity

DDFC's level of complexity is related to what makes it innovative: the incorporation of players' heartbeats in core gameplay mechanics. Players will need to wear a Fitbit smartwatch when playing DDFC. Fitbit will detect the player's HR, and the data will be collected and transmitted via the Fitbit Web API. Once DDFC records the HR data, it will translate the data into audio that plays on the enemy player's client. Furthermore, the audio levels will vary depending on how fast the player's HR is and their proximity to the enemy player.

In addition to the complexity of marrying HR data with in-game audio, DDFC will be a networked game. The developer is familiar with the Unity game engine and the programming language C#, with which they will build DDFC. However, the networking component will be uncharted territory. DDFC will use Mirror Networking (Mirror) for its multiplayer aspect, which will also need to communicate with the Fitbit Web API. Mirror is a library designed to abstract the networking logic so that developers can put more attention to developing the game. It is a "high level Networking library for Unity, optimized for ease of use & probability of success" (Mirror Networking, n.d.). Mirror supports the Telepathy Transport library, which uses the Transport Control Protocol (TCP). Because of its use of TCP, Telepathy is fast and reliable; it ensures that essential data such as the player's state (HR, Health, etc.) will be reliably transmitted. Mirror supports User Datagram Protocols (UDP) as well, so there are alternatives to

Telepathy if it is discovered that utilizing UDP for its speed is more necessary than the reliability of a TCP connection.

10. Technical Challenges

Biometric Data Translation
DDFC will be the developer's first opportunity to utilize biometric devices during the development of an application. The developer must learn to connect a Fitbit to the game application in Unity, read each player's HR data, then convert the data into in-game audio as close to real-time as possible.

Development of a Multiplayer Game
Although the developer is familiar with the Unity game engine and has worked on a multiplayer game project, she was not part of the networking team. Creating a multiplayer game, using Mirror, and communicating with a JavaScript Web API with Unity will be new experiences.

11. Development Schedule and Milestones

This development schedule and timelines were created under the assumption that the project's start date will be January 3, 2023, and the term is fifteen weeks long. Note that testing and debugging will occur during every milestone, so the following table does not explicitly mention them. The format of the Timeline is **MM/DD**. The Breakdown and Total columns use **hours** as the unit of measurement. **Breakdown describes the estimated number of hours** dedicated to a particular milestone, and **Total describes the cumulative hours** spent working on the project to date.

(Development Schedule Table Continues on the Next Page.)

Timeline	Tasks	Breakdown	Total
	Milestone 1		

01/03–01/24	• Fitbit API Connectivity
	  ○ User authorization
	  ○ Get player's Heart Rate data
	• Player
	  ○ Class/Initial State
	  ■ Health
	  ○ NetworkVariables (class)
	• Map
	  ○ Empty 3D room
	• Network
	  ○ End-to-end server connectivity
	  ○ Matchmaking (players can connect)
		84	**84**

	Milestone 2		

01/25–02/07	• Fitbit API Connectivity
	  ○ Translate Heart Rate data into serializable data (to send across the server)
	• Player
	  ○ Movement
	  ○ Attack
	• Combat System
	  ○ Weapon
	  ○ Give/Take damage
	• Map
	  ○ Initial design/layout (greyscale)
		60	**144**

	Milestone 3		

02/08–02/21	• Network
	  ○ Send & receive data
	  ■ Players' health
	  ■ Players' movement
	  ■ Hit registry
	• Asset Management
	  ○ Import Textureless 3D Models
		50	**194**

	Milestone 4		

02/22–03/14	• User Interface (UI)
	• Fitbit API Connectivity
	  ○ Deserialize Heart Rate data, then turn into audio cues
	• Asset Management
	  ○ Lighting
	  ○ Textures
		72	**266**

	Milestone 5		

03/15–03/28	• UI
	  ○ Settings page
	  ○ Stats page
	• Player Stats Storage
	  ○ Read/write to persistent data
	• Asset Management
	  ○ Sound design
	  ■ Sound effects
	  ■ Background music
		32	**298**

	Milestone 6		

03/29–04/04	• Balancing and Polishing
	  ○ Map design
	  ○ Player stats (health vs. attack, TTK, etc.)
		42	**340**

	Milestone 7		
04/05–04/18	• Final testing and polishing • Generate Final Report	20	**360**

12. Deliverables

The final deliverables for the project are as follows:

- Unity project files (and C# source code) via GitHub
- Executable Application that runs on Windows and macOS
- README file containing information regarding tools and frameworks (and their versions) and how to setup/run the project files in Unity
- Testing documentation
- Major Project Final Report

13. Conclusion and Expertise Development

Through CST's Technical Programming Option and the first year of BTech, the developer has built a foundation of skills for game development. Thus far, the developer has experience working on single-player games. Through the development of DDFC, she will gain the knowledge and skills to make and understand multiplayer games. While basic Unity functionality and C# are familiar, the developer will be utilizing new frameworks and tools to further her expertise.

Unity and C# are used by many development studios, so working on DDFC will sharpen the developer's existing proficiency in addition to providing deeper knowledge and insight. Creating a multiplayer game provides value, as possessing networking skills will provide a broader range of opportunities and advantages. After the development of DDFC has been completed, the developer will have attained valuable skills that will be desirable to future project teams and employers.

14. References

Bateman, C., Lowenhaupt, R., & Nacke, L. E. (2011). *Player Typology in Theory and Practice*. Authors & Digital Games Research Association DiGRA. Retrieved October 23, 2022, from http://www.digra.org/wp-content/uploads/digital-library/11307.50587.pdf

Faller, P., & Babich, N. (2019, December 17). *What Are User Personas and Why Are They Important? | Adobe XD Ideas*. Adobe XD | Fast & Powerful UI/UX Design & Collaboration Tool. Retrieved October 23, 2022, from https://xd.adobe.com/ideas/process/user-research/putting-personas-to-work-in-ux-design/

Game-Ace. (2021, July 19). *How to Make Low-poly Models for Games*. Game-Ace. Retrieved October 22, 2022, from https://game-ace.com/blog/low-poly-models/

Hanna, T. (n.d.). *What is biometrics?* TechTarget. Retrieved October 19, 2022, from https://www.techtarget.com/searchsecurity/definition/biometrics

Hunicke, R., LeBlanc, M., & Zubek, R. (2004). MDA: A Formal Approach to Game Design and Game Research. 5. https://users.cs.northwestern.edu/~hunicke/MDA.pdf

Kumar, J. M., Herger, M., & Friis, R. (2020, June 5). *Bartle's Player Types for Gamification*. Interaction Design Foundation. Retrieved October 22, 2022, from https://www.interaction-design.org/literature/article/bartle-s-player-types-for-gamification

Mirror Networking. (n.d.). *Mirror Networking*. Mirror Networking - Mirror. Retrieved November 17, 2022, from https://mirror-networking.gitbook.io/docs/

The Myers & Briggs Foundation. (n.d.). *MBTI® Basics*. The Myers & Briggs Foundation. Retrieved October 23, 2022, from https://www.myersbriggs.org/my-mbti-personality-type/mbti-basics/

sgc908. (2022, March 27). *Agile software development technique-I* - [Photograph]. Kharbari.com. Retrieved October 13, 2022, from https://kharbari.com/agile-software-development-technique-i/

Thomét, M. (2012, January 22). *Player Taxonomies: Reviewing the DGD1 Model | Set in Cobalt*. Set in Cobalt. Retrieved October 23, 2022, from https://incobalt.wordpress.com/2012/01/22/player-taxonomies-reviewing-the-dgd1-model/

15. Change Log

The following table will describe all changes made to the document. The Date column (formatted as **MM-DD-YYYY**) indicates when the change was made.

Date	Version	Description
11-05-2022	1.0	Submitted version 1.0 to the committee
11-19-2022	2.0	Added Section 7.1. (p. 18-19) to describe the algorithm for HR data translation to audio levels. Update Section 3 (p. 11) to describe what will occur if the player's

		HR is not detected. Updated Sections 6.2. (p. 16), 9 (p. 19-20), and 10 (p. 20). Mention the use of “Mirror Networking” instead of “Netcode.” Mirror uses TCP (instead of Netcode’s UDP); this addresses the concern of reliable data transfer.