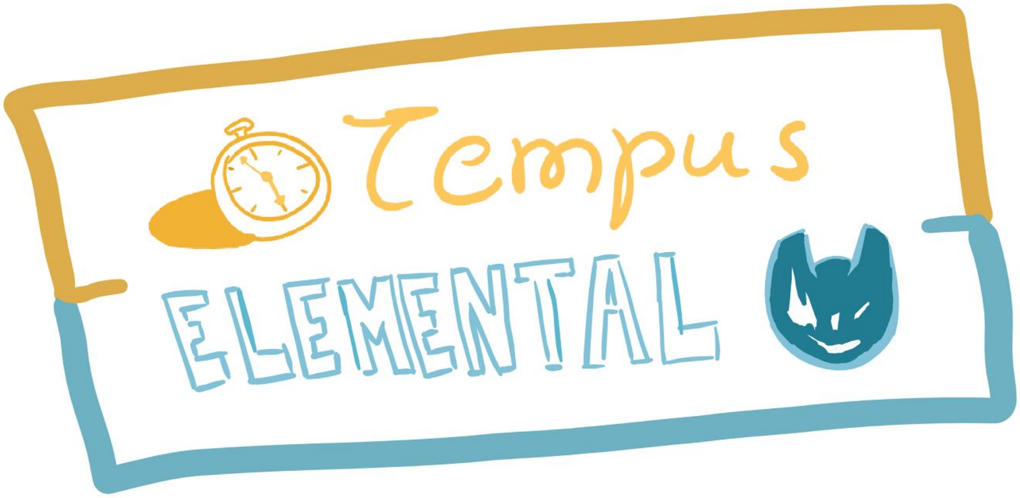


Tempus Elemental

Design Document
09.15.2017

Team 1
Harrison Chen
Han Wang
Mehran Einakchi
Elijah Hauber
Michael Parr
Brian Rhee

Index

Purpose

Almost every combat game nowadays adheres to the same old used up formula of HP-Bar, Mana-Bar, and GP-Count. We're here to break that mold and shoot for a new concept that will spice up the gaming atmosphere by twisting that concept in a new direction.

This project is a 2D battle arena game that uses "time" and its manipulation as the core mechanics to create a fun and engaging experience. It revolves around the idea of a top down co-op arena combat game that combines fast paced action with meticulous tactics and execution. In the gameplay, some of the more unique and markable aspects include but are not limited to, manipulation of time during combat, time as a form of player health and player actionable-guage, and responsive and fun set of weapons or actions to use when playing in co-op together, which all tie together thematically.

Tempus Elemental will provide the user with a gameplay that is memorable. The players can truly indulge in the game by being provided simplicity through our unique time mechanic.

Functional Requirements

<u>Users can fight each other in unique ways</u>

As a player,

1. I would like to chain melee attacks into a combo for more dynamic gameplay.
2. I would like to have a ranged attack, that costs time to "throw."
3. I would like to charge my attack to use the ranged throw, this would stop movement.
4. I would like to be able to aim while charging the attack.
5. I would like to have a backstab mechanic in which a player can inflict/take more time when meleeing another player from behind.
6. I would like to be able to dash through attacks/arrow from ranged weapons.
7. I would like to distort time in a radius around me to add a twist/challenge to those fighting me (costs extra time).
8. I would like to own a large quantity of seconds to use as currency.

<u>Users can use powerups to create challenges</u>

As a player,

1. I would like to pick up power-ups that allow me to further explore the idea of time, by permanently changing my time distortion mechanic.
2. I would like the default time distortion to slow the time around me.
3. I would like to have a power-up that changes my distortion to speedup time.
4. I would like to have a power-up that changes my distortion to freeze time around me.
5. I would like consecutive identical distortion power-up pickups to multiply the effect of the distortion.
6. I would like a powerup that slows time.
7. I would like a powerup that changes my distortion to rewind time.
8. I would like a one-time pickup that increases my remaining time.
9. I would need to steal "time" from other players, creating competition and survival instincts to occur between players.

Users can determine where they are in the game visually and lifewise

As a player,

1. I would like a HUD text indicator to show my time remaining.
2. I would like a radial circle to indicate the time remaining.
3. I would like to differentiate myself from the people I am facing visually using customization options or character options.
4. I would like to know when I am about to die (via visual indicator).

Users can play with friends and/or strangers

As a player,

1. I would like to play with my friends.
2. I would like to play with at least three friends.
3. I would like to play against my friends or strangers.
4. I would like to switch teams between players.

Users can select different modes to play with

As a player,

1. I would like many different modes in the case of a tie (customizable by the players)
2. I would like the base standard for the game to end in a draw if there is a tie.
3. I would like a sudden death mode with less time than standard as well as the removal of the time steal mechanic.
4. I would like to have a option menu for game mode values.
5. I would like to vote for the next game mode/map.
7. I would like a game mode that pits players together in a deathmatch free for all, last player standing wins.(FFA)
8. I would like a game mode that pits two teams against each other in a deathmatch to run the other team's cumulative time out. (TDM)
9. I would like a game mode in which players play to capture the most time from different points on the map in a king of the hill style game. (KOTH)
10. I would like a game mode that causes stolen time to instead fill up a U.I. bar in the player's color. The player who contributed the most after the bar is filled wins, infinite respawns (after timeout). (Vampire Mode)

11. I would like to be able to select game modes that has players capture flags to homebase. (CTF)
12. I would like a vanilla gamemode FFA without any powerups.

<u>User may be pleasantly surprised when these features are implemented</u>

If time allows,

1. I would like to play an adventure mode that has a story line.(Adv)
2. I would like to have a lobby to organize online play.
3. I would like to be able to integrate online and local play.
4. I would like a reward for successive wins.
5. I would like to play the game on different platform.
6. I would like to play with/against with players from different platform.
7. I would like the game be able to record my match history.
8. I would like to save my single player game process.
9. I would like to use time as a currency to buy rewards, such as randomized cosmetic items.
10. I would like to pair against strangers online.
11. I would like to pair with strangers online.

Kits:

1. I would like to choose between multiple characters with different “kits.”
2. I would like to have each character look aesthetically different.
3. I would like to have a character that starts with more time, but moves a bit slower.
4. I would like to have a character that moves faster, but steals less time from other players.
5. I would like to have a character that steals more time from other players, but starts with less time.

Non-Functional Requirements

1. As a player, I would need to move consistently and one to one with my inputs with lowest input delay possible. (Response time)
2. As a player, I would like the game's interface to be easy to pick up but difficult to master. (UI)
3. As a player, I would like to have a melee attack that feels like an attack that has impact; action and reaction sort of animation or visual effects.
4. As a player, I would like the game to be very responsive(low response time) because it's a real time video game.
5. If time allows, I would like the game to have a wide selection of maps.
6. If time allows, I would like the game to have a custom map creator.
7. If time allows, I would like the game to have an adventure co-op mode.
8. If time allows, I would like the game to have war fog or limitations on vision of the player.
9. If time allows, I would like the game to have a floodlight/blind mode of gameplay mixed with a split screen form of gameplay.
10. If time allows, I would like the game to have more than one game mode or custom lobby type.
11. If time allows, I would like the game to have a sparklight mode of gameplay where the player can only see around the character when the character performs an attack on a nearby object or player.
12. If time allows, I would like the game to look sleek and smooth in its animations and art style.
13. If time allows, I would like the music and sound effects to really immerse me in the game.
14. If time allows, I would like the game to have scalability of the UI for different platform. (Usability)
15. If time allows, I would like the game to have colorblind mode.(Accessibility)
16. If time allows, I would like the game sessions and player files to be safe and secure so that the game is not vulnerable to hacking. (Security)

Design Outline

This project will be a game that allows players to battle each other in a confined space within the time limit, explore the map and discover Power-Ups. Each player's life/health is determined by their own "Biological Clock", and it keeps ticking down until player runs out of time/health then dies. The last player standing wins. This game will use a PC as the local client mode, and players will control their characters using controllers as input method.

Components

1. Client
    a. The game will be based on Unity which provides a physics engine, UI elements, and 2D graphic rendering.
    b. The client will show the user a menu to change settings, select game modes, and begin the game.
2. Menu
    a. This component can change things about the game in settings like audio volume, co-op or vs., color blind mode, etc.
    b. Game modes will be separate from settings since this will be interacted with more by the user.
    c. Game modes will select types of game modes like fill the bar (FTB), king of the hill (KOTH), team deathmatch (TDM), etc.
    d. The user will be able to start a new game from the menu, after choosing a gamemode and map.
3. Game
    a. Map
        i. The selected map will have different thematic elements, as well unique designs (for example, rotating walls)
        ii. The map will spawn powerups in key locations over time.
    b. Game Controller
        i. This component will keep track of/spawn each player, and control the logic of the game.

   ii. After this component declares a winner and notifies the user, it will transition the game back to the menu.

c. Player
   i. Controller Input
      1. The controller will retrieve user input and translate the input into a defined action.
      2. The input can be one-to-one or one-to-many depending on the sequence, combination, and/or buttons pressed.
   ii. Each player will have a set amount of time left, used as both health and "stamina/mana"
   iii. Players can create Time Distortions in a radius around them
   iv. Players instantiate a variety attacks to steal time from other players

d. Attacks
   i. Each attack has its user and if it collides with a target, the target will lose time on their behalf, as a pseudo life point.
   ii. The attack will coordinate the transfer of time from one player to another, so the target loses time and the equivalent time lost is given to the attacker.
   iii. Attacks interact with each other by projectile and melee attacks. These attacks are not hitscan (meaning immediate) and have a travel time and travel distance to any target in its path. This provides a chance for players to duke it out with dodges and good timing
   iv. In this way, the target player has to try and dodge attacks, as well as the attacker having to decide when or where to launch an attack.
   v. The projectile attack is an elemental power bullet that shoots out similarly to a crossbow.
   vi. The melee attack is similar to that of a katana swing.

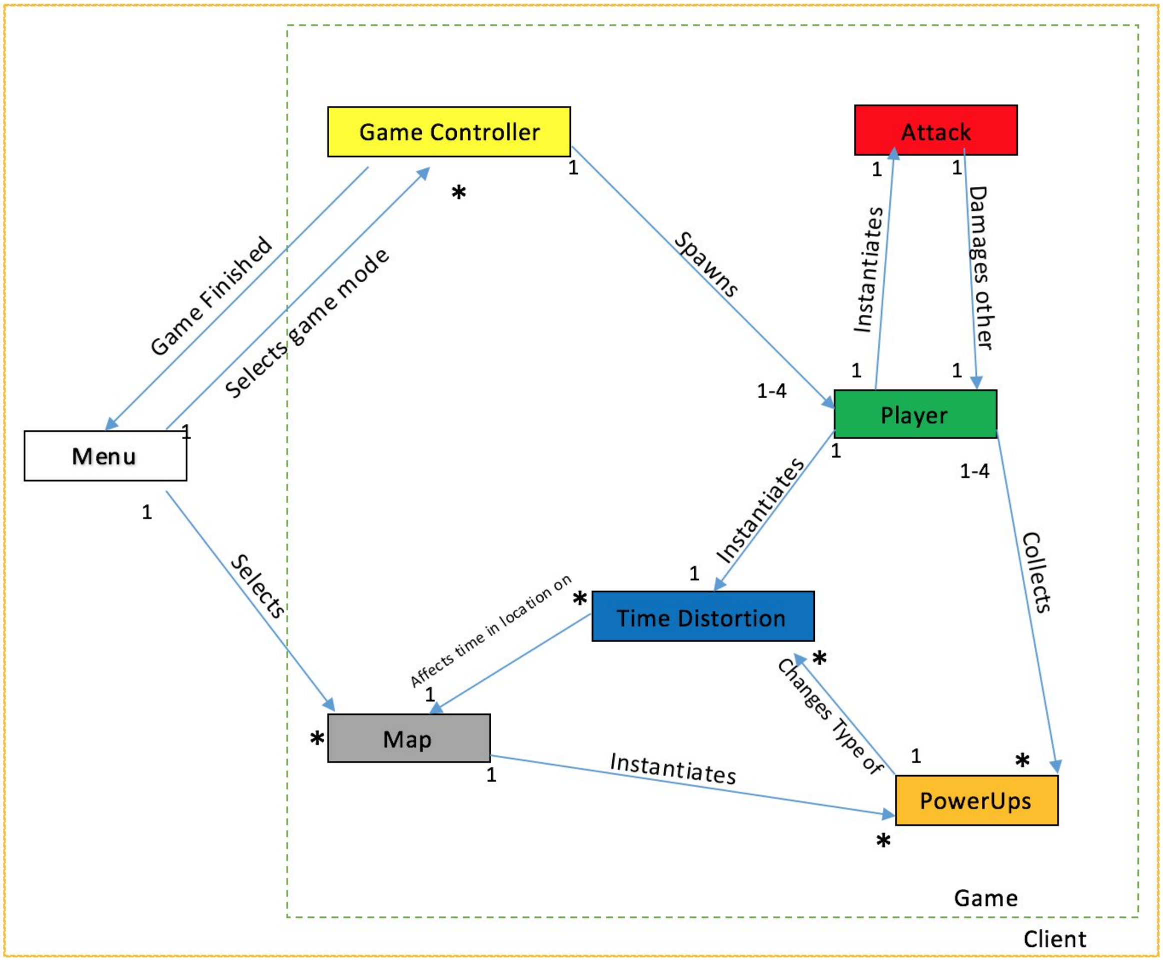
e. Time Distortion
   i. Interacts with every object within its radius on the map

- ii. Power-ups collected by the player will change the type of Time Distortion that the player can instantiate
- iii. Various types
    1. Slow down
    2. Speed Up
    3. Freeze
    4. Reverse
- iv. When multiple distortions collide, the area of collision combines the two distortions

f. Powerups

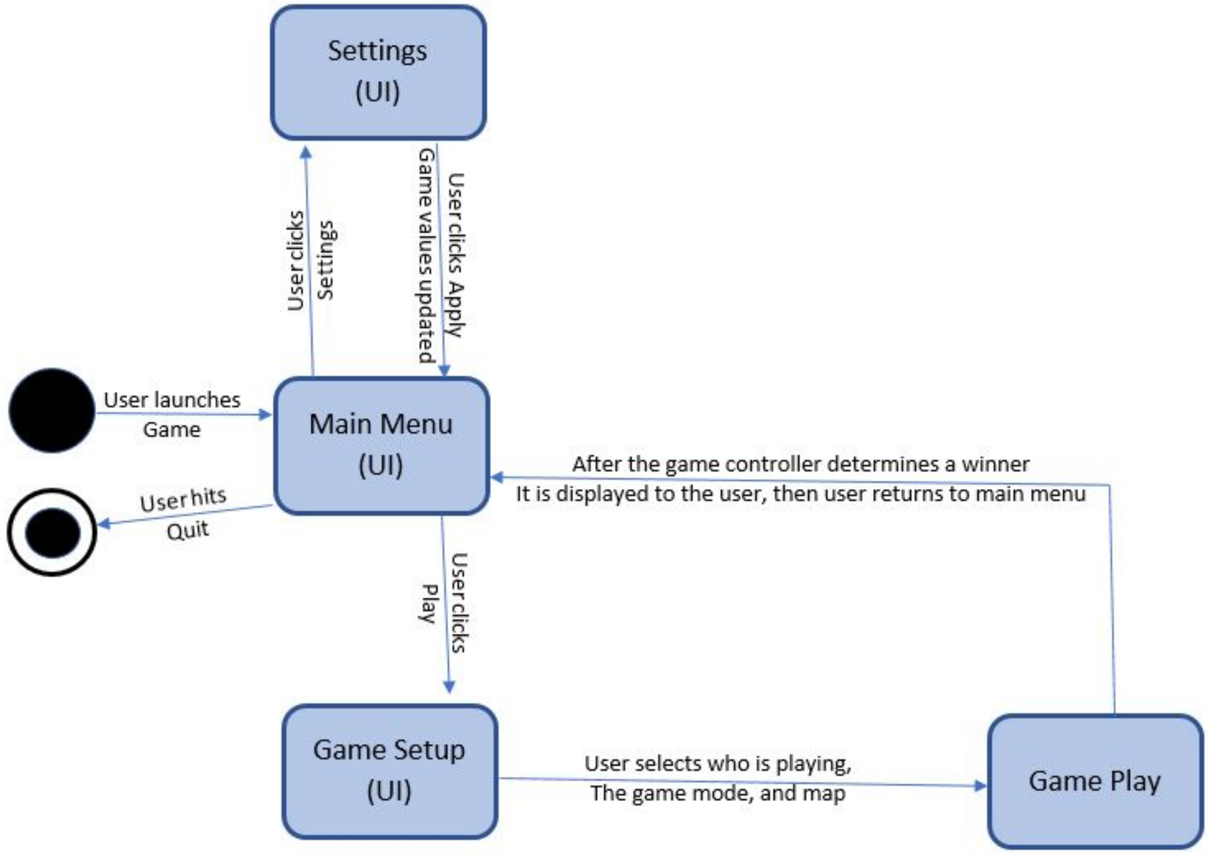
- i. These are spawned across the map for players to pick up and collect
- ii. Each powerup is tied to a type of time distortion
- iii. Players will instantiate that the power-up’s type of distortion upon pickup.

High-Level Overview:



Our game must provide the user with options to enhance their enjoyment. The user is going to start the game at the menu. From the menu they will have the choice to change settings to their liking. After that, the game begins. Players then compete with each other by trying to steal time from one-another. Time is stolen through the use of attacks. Additionally, each player can “spend their time” to distort the flow of time in an area around them. This distortion is determined by power-ups collected, which are spawned by the map. The game controller oversees everything that happens and determines when there is a winner.

State/Activity Diagram:



The Client's activity state will either be in a game or in the menu. There are three different menus that the user can click through. From the Game Setup menu, the user can then create a game to join. Depicted as one block above, there are many different kinds of games that the user can select.

Design Issues

Functional issues

How should the ranged attack be implemented?

Option 1: Melee attack button turns into a ranged attack after specified time held

Option 2: Separate button that implements the ranged attack

Choice: Option 1

Discussion: By mapping the charged ranged attack to the same button as the melee

attack, it simplifies the amount of buttons the user has to remember, and adds reusability to the inputs of the player while enhancing the learnability of the game.

What happens when projectiles collide?

Option 1: Both/All projectiles disappears/ Cancel out

Option 2: Projectiles ricochet based on their own velocity and vector.

Option 3: Projectiles pass through each other and do not affect each other.

Choice: Option 3

Discussion: While having projectiles ricochet off each other adds additional strategy to the game, it also increases the complexity of each projectile. Under the time constraint, we have decided to go for option 3 because it does not effect, but if we have extra time at the end, we can come back to Option 2.

Multiplayer Input

Option 1: Each player has a client

Option 2: Each player has a controller

Choice: Each player has a controller

Discussion: Forcing each player to own their own client (computer & copy of the game) goes against our goal of couch multiplayer. We want part of the fun to simply come from being in close proximity to friends. This typically ends up being cheaper overall for the user.

<u>What to do after there is a tie</u>

Option 1: There is a draw as a standard and the users can choose a different mode in the case that there is a tie
Option 2: Sudden Death
Choice: Draw
Discussion: A draw is a good standard. Many people may not want to have a separate sudden death game mode that takes away from the main game mode that they played to get to a tie.

<u>Where to display the Time Counter</u>

Option 1: Display the Time Counter in each corner for each player
Option 2: Change the transparency of the player depending on how much Time is left
Option 3: Display the Time counter above the players' heads
Choice: All of the above!
Discussion: Implementation of all three options are relatively simple. Having a standard
HUD (Heads up display) time counter is essential and having the counter above the heads helps to keep the game immersive.

Non-functional issues

Which game engine is best for creating our game?:

Option 1: Game Maker

Option 2: Unreal Engine

Option 3: Unity3D

Choice: Unity3D

Discussion: Unity allows us to use C#, and the license for this project is free.

Additionally, some of the developers have prior experience with the engine. Finally, since the physics engine is already implemented, it allows the developers to spend more time creating game mechanics rather than back end classes.

How can player tell how much time each others have?

Option 1: Each player has their own Heads Up Display (HUD) in each corner of the screen.

Option 2: Digital display for (health) on top.

Option 3: RGB color scheme for player indicating their time left, Players become more translucent as their time expires

Option 4: Tiny clock around players that indicating time.

Choice: A combination of options 1, 3, and 4.

Discussion: By having multiple ways of showing the user how much time they have left, it increases the usability and learnability of users

Should the camera perspective be top-down or from the side?

Option 1: Top-Down

Option 2: Side-View

Choice: Top-Down

Discussion: While top-down games are simpler to implement, side-view games add an aspect of verticality to the game. To implement a side-view perspective, each player will need some way to jump up to higher levels. We have decided to go with Top-down because of limitation of time. We will probably develop on side-view later on or we have more time.

Which language is appropriate for creating scripts in unity?
Option 1: JavaScript
Option 2: C#
Option 3: Boo
Choice: C#
Discussion: C# is one of the most compatible languages with Unity engine, since we have a lot of experience working with C and C++, and C# is not too far apart from the C language family, which makes it easier for us to adapt to it.

What genre of audio would add the most immersion to our game?
Option 1: Techno
Option 2: Orchestral
Option 3: Light Jazz or Bebop Jazz
Option 4: 8-bit
Choice: We choose to opt for an electronic or synthetic fast paced audio track.
Discussion: We choose this style because it provides a rush of adrenaline from the audio perspective that can enhance the fast paced gameplay of making quick reactions and decisions since this is primarily an action based game.

What style of art should we use?
Option 1: Pixel art
Option 2: Cartoonish
Option 3: 3D Round Shapes
Option 4: Realism
Option 5: Vector/Line Art
Option 6: 3D Polygonal

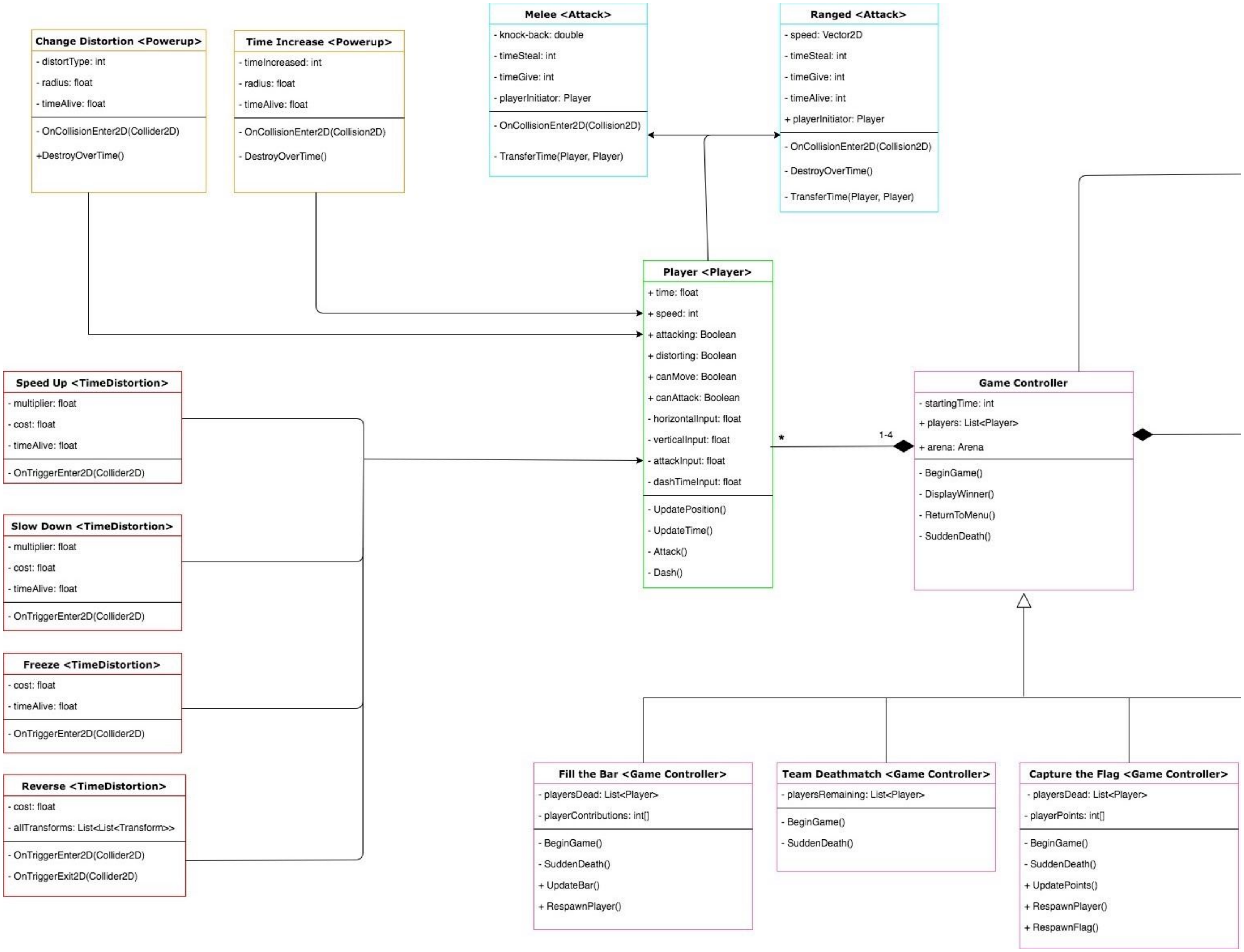
Choice: We Choose to use Option 2, Cartoonish Art Style
Discussion: We choose this because it is an art style that is pleasant to look at and feasible to draw and make quick edits without using overly convoluted rendering or drawing methods.

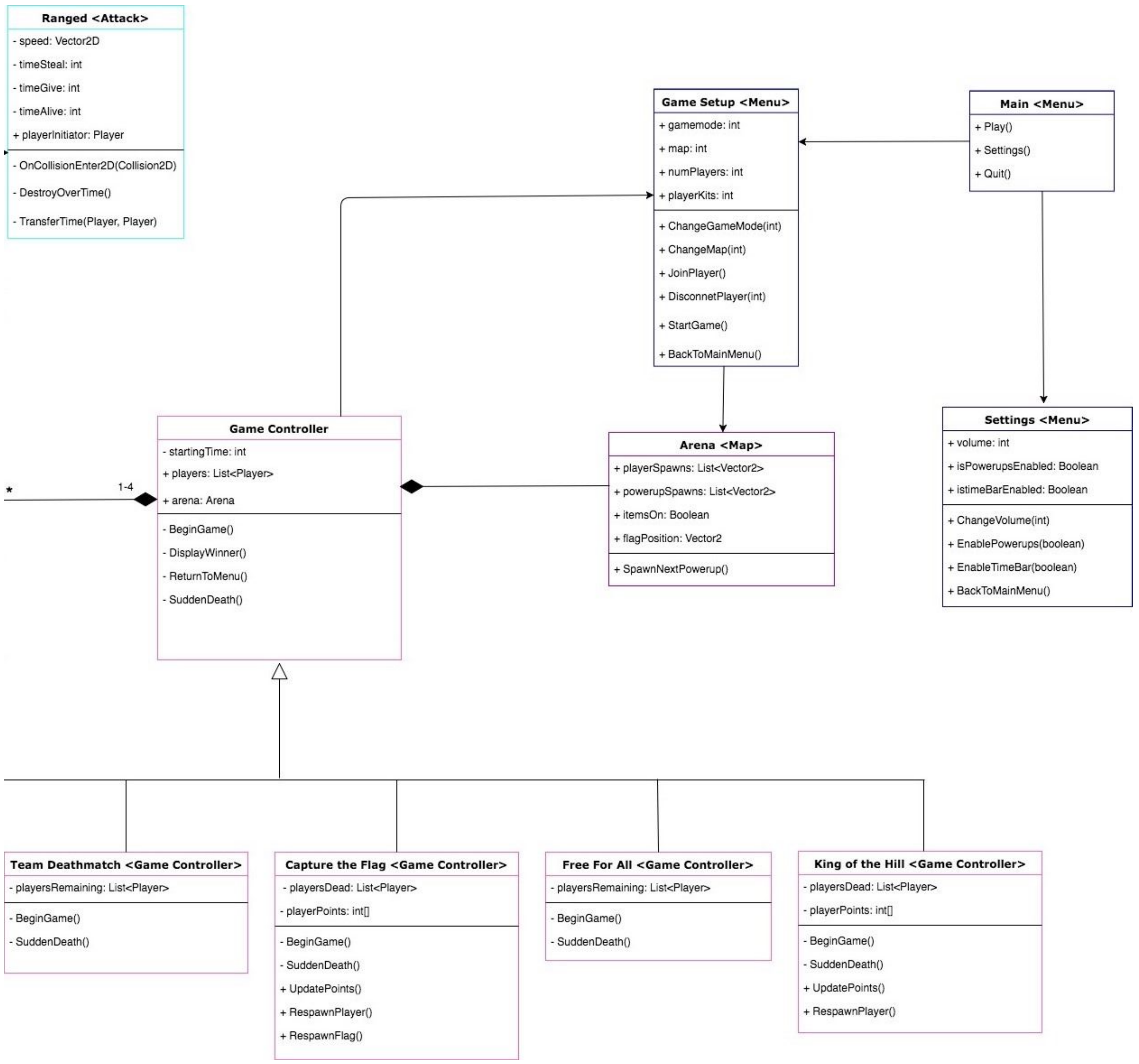
<u>What style of animation should we use?</u>

Option 1: Hand draw each frame
Option 2: Use key frame animations
Choice: Key frame animations
Discussion: While hand drawing each game is typically better received, it takes an extremely long time to do. Additionally, Unity's key frame animations can call public functions. This reduces the need for embedded timers in scripts.

Design Details

Class Level-Design Diagram (next page):


Change Distortion <Powerup>
- distortType: int
- radius: float
- timeAlive: float
- OnCollisionEnter2D(Collider2D)
+DestroyOverTime()
Time Increase <Powerup>
- timeIncreased: int
- radius: float
- timeAlive: float
- OnCollisionEnter2D(Collision2D)
- DestroyOverTime()
Melee <Attack>
- knock-back: double
- timeSteal: int
- timeGive: int
- playerInitiator: Player
- OnCollisionEnter2D(Collision2D)
- TransferTime(Player, Player)
Ranged <Attack>
- speed: Vector2D
- timeSteal: int
- timeGive: int
- timeAlive: int
+ playerInitiator: Player
- OnCollisionEnter2D(Collision2D)
- DestroyOverTime()
- TransferTime(Player, Player)
Player <Player>
+ time: float
+ speed: int
+ attacking: Boolean
+ distorting: Boolean
+ canMove: Boolean
+ canAttack: Boolean
- horizontalInput: float
- verticalInput: float
- attackInput: float
- dashTimeInput: float
- UpdatePosition()
- UpdateTime()
- Attack()
- Dash()
*
1-4
Game Controller
- startingTime: int
+ players: List<Player>
+ arena: Arena
- BeginGame()
- DisplayWinner()
- ReturnToMenu()
- SuddenDeath()
Speed Up <TimeDistortion>
- multiplier: float
- cost: float
- timeAlive: float
- OnTriggerEnter2D(Collider2D)
Slow Down <TimeDistortion>
- multiplier: float
- cost: float
- timeAlive: float
- OnTriggerEnter2D(Collider2D)
Freeze <TimeDistortion>
- cost: float
- timeAlive: float
- OnTriggerEnter2D(Collider2D)
Reverse <TimeDistortion>
- cost: float
- allTransforms: List<List<Transform>>
- OnTriggerEnter2D(Collider2D)
- OnTriggerExit2D(Collider2D)
Fill the Bar <Game Controller>
- playersDead: List<Player>
- playerContributions: int[]
- BeginGame()
- SuddenDeath()
+ UpdateBar()
+ RespawnPlayer()
Team Deathmatch <Game Controller>
- playersRemaining: List<Player>
- BeginGame()
- SuddenDeath()
Capture the Flag <Game Controller>
- playersDead: List<Player>
- playerPoints: int[]
- BeginGame()
- SuddenDeath()
+ UpdatePoints()
+ RespawnPlayer()
+ RespawnFlag()


Ranged <Attack>
- speed: Vector2D
- timeSteal: int
- timeGive: int
- timeAlive: int
+ playerInitiator: Player
- OnCollisionEnter2D(Collision2D)
- DestroyOverTime()
- TransferTime(Player, Player)
Game Setup <Menu>
+ gamemode: int
+ map: int
+ numPlayers: int
+ playerKits: int
+ ChangeGameMode(int)
+ ChangeMap(int)
+ JoinPlayer()
+ DisconnetPlayer(int)
+ StartGame()
+ BackToMainMenu()
Main <Menu>
+ Play()
+ Settings()
+ Quit()
*
1-4
Game Controller
- startingTime: int
+ players: List<Player>
+ arena: Arena
- BeginGame()
- DisplayWinner()
- ReturnToMenu()
- SuddenDeath()
Arena <Map>
+ playerSpawns: List<Vector2>
+ powerupSpawns: List<Vector2>
+ itemsOn: Boolean
+ flagPosition: Vector2
+ SpawnNextPowerup()
Settings <Menu>
+ volume: int
+ isPowerupsEnabled: Boolean
+ istimeBarEnabled: Boolean
+ ChangeVolume(int)
+ EnablePowerups(boolean)
+ EnableTimeBar(boolean)
+ BackToMainMenu()
Team Deathmatch <Game Controller>
- playersRemaining: List<Player>
- BeginGame()
- SuddenDeath()
Capture the Flag <Game Controller>
- playersDead: List<Player>
- playerPoints: int[]
- BeginGame()
- SuddenDeath()
+ UpdatePoints()
+ RespawnPlayer()
+ RespawnFlag()
Free For All <Game Controller>
- playersRemaining: List<Player>
- BeginGame()
- SuddenDeath()
King of the Hill <Game Controller>
- playersDead: List<Player>
- playerPoints: int[]
- BeginGame()
- SuddenDeath()
+ UpdatePoints()
+ RespawnPlayer()

Description of Classes and their Interactions

Unity Scripting Engine:

Every script (class) in Unity is attached to a GameObject. Unity provides the life cycle methods such as *start(), awake(), update(), onDestroy()....* Etc. for every script. Code in Unity involves this life cycle, especially methods like *update()* -- which is called every frame and *start()* which is called once when the scene is loaded. Thus, every class in our class diagram also has the start and update methods.

Menu:

- Represents the start-up menu to navigate and change different configuration for the game and game modes.
- It includes subclasses like Game Setup, Settings.
- Game Setup stores the value of game modes\maps\number of players and players' choice of kits and functions to modify these variables.
- Setting stores the value of Game Volume\Disabling and enabling of Powerups and Time bars and functions to modify them.

Arena:

- Represents the selection of maps that player can choose from.
- Maps are created by developer, the environments are static and have custom spawn points for players and powerups.

Game Controller:

- Game Controller is construct with a parent class "Game Controller" and subclasses including "Fill the Bar" "Team Deathmatch" "Capture the Flag" "King of the Hill" and "Free For all".
- The parent class "Game Controller" share its variables like PlayerList or arena with children classes.
- The parent class contains the main function that begins the game and is able to Display winner and go into Sudden Death mode when certain conditions are met.

- The children classes extend the “Game Controller” class and differentiate them from one and another by having different fields of conditions to decide the winner.

Player:

- The player initiates attacks both melee and ranged.
- They receive attacks both melee and ranged based on whether the attack collides with the player boundary.
- On top of initiating and receiving attacks the players can distort time that changes the speed of other players and projectiles.
- Upon running out of time, the player will die by destroying itself and notifying the game controller.

Melee Attacks:

- Melee attack will knock-back and steal time from the opposing player.
- After an attack lands on an opposing player the player who initiated the attack gains the time stolen.
- Stops the player from moving while in action.

Range Attacks:

- Ranged attacks will destroy themselves over time after being successfully initiated.
- Ranged attacks are “charged”, meaning the player will have to hold a button to “wind them up”.
- After the attack lands it performs similarly to melee attacks. The initiating player gains time while the opposing player loses time.
- Stops the player from moving while aiming, but the player can still turn.

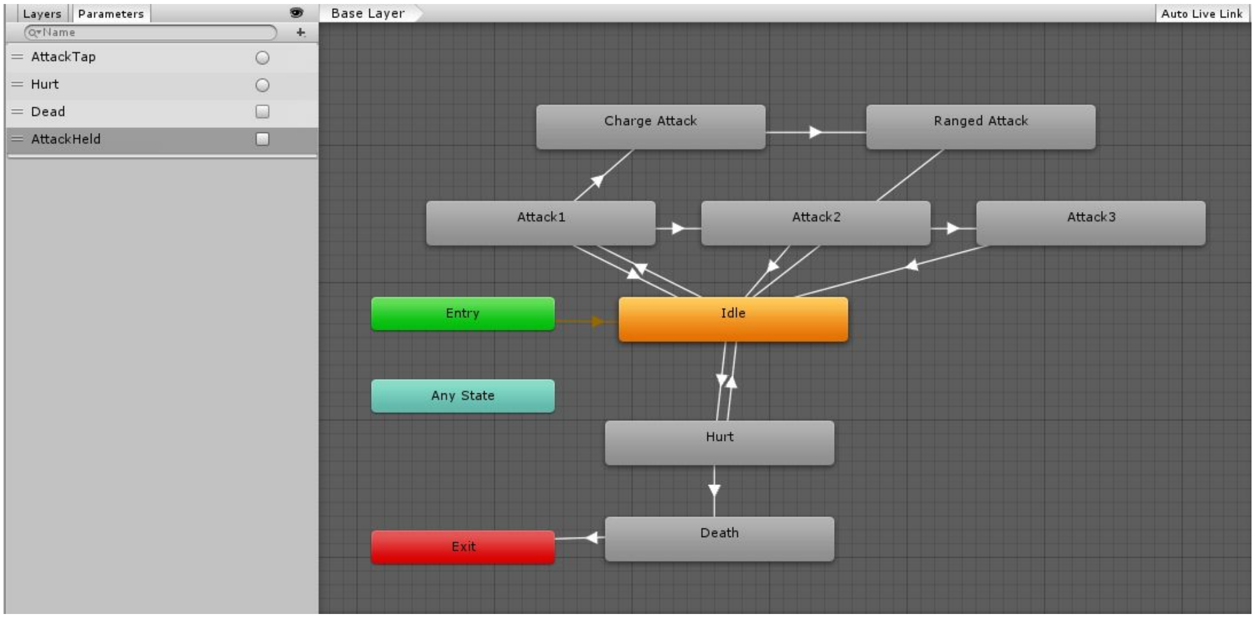
Powerup:

- Each powerup can be collected by a player.
- Once collected, that player’s time distortion is changed.
- There is also a powerup that gives the player more time.

Time Distortion:

- Creates an area around the player that affects the flow of time.
- Every object (Player, Attack, etc.) that enters the area will be affected.
- Affected object’s speed will either sped up, slowed down, frozen, or reversed.

Player Animation Diagram:



The player character has many different animation states that must be transitioned through, in order to give the player visual feedback on the actions he/she is performing. The orange Idle animation acts as homebase, from which every other animation should start and end, barring death where the player will disappear by the end. The AttackTap trigger is used when the attack key is pressed, and when pressed consecutively it will go along the attack 1, 2, and 3 combo.

If not pressed the player will go back to idle at any point in the combo. If the attack button was instead held, the AttackHeld boolean will be true, thus Attack1 will instead transition to the Charge Attack. This charge animation will then loop until the button is released, where the ranged attack animation will play before returning the player to idle. If the character is hurt while not doing something like attack (which takes precedence), the player will go through an animation.(Additionally, the low-health player's body gets more transparent the less health they have left) From there, if the character had died, they will proceed to the death animation, otherwise the character will return to idle.

Game Modes:

Vampire Mode

This mode is a gamemode where all the players are playing to steal as much time from the others as possible. In the end of the round, whoever has stolen the most amount of time from the other players, wins as the vampire of the match. Thus, the name Vampire Mode.

Beast Mode

Beast Mode is a mode where one player, chosen at random, is spawned into the arena with an elevated amount of power, speed, size, and time-gauge. This essentially creates a super being where all of the other players on the map are now teamed up as one to fight against the larger evolved form, duking it out to see if the monster or the smaller players win.

Vanilla Mode

This mode is simply the regular mode of the game, where everyone plays in a free for all brawl until only one player is left standing. This is a very simple game mode and should be understood as just that.

Capture The Flag

CTF is where all the players on the screen are split into teams or set as a free for all where people attempt to go for a flag that they bring back to their spawn zone, except the players are also capable of using their elemental attacks to prevent the other players from completing their objective, such as a temporal seed to slow down the player with the flag or using an elemental charge bullet to eliminate the player who is holding the flag.

TDM

Team Death Match is the mode that is mechanically similar to Vanilla Mode, with the exception of having a set of teams going at each other instead of free for all. In this mode, players are also allowed to use temporal seeds to resurrect down teammates or sacrifice the rest of their own time-life to resurrect or boost the abilities of a nearby teammate. This sort of game mode brings a new flavor to the game and encourages players to be playing with a more selfless and exciting mindset.

KOTH:

King of the Hill is a game mode that can be played as free for all or team based, where the objective is to capture and hold a point on the map for as long as possible, and whenever the player capturing and holding the point is continuing to hold the point, the player receives a slowed down rate of losing their time-life. The opposing team attempts to fight against the team that is capping a point to attempt to cap and hold the point for their own team and come out as the winner when the global timer of the match ends.

Modifiers: (These can be applied to most of the game modes above)

Competitive Mode (No Power-ups)

This means that during the game, no one is allowed to use powerups, temporal seeds, or other misc. Pickups. This is to shave down the gameplay to giving the players only the character’s movement and character’s attacks, attempting to create an environment where the player with the most skill wins, rather than relying on powerups to “cheat” your way to victory. This takes out some of the excitement of the game for a more streamlined and fair set of gameplay for those who seek this sort of gameplay.

Melee Only

As the name indicates, all players in the game are only allowed to use their melee weapons during this game mode. This provides a game style that encourages players to be in closer proximity and try to play a game of slashing back and forth to victory, like a fencing match with slo mo effects.

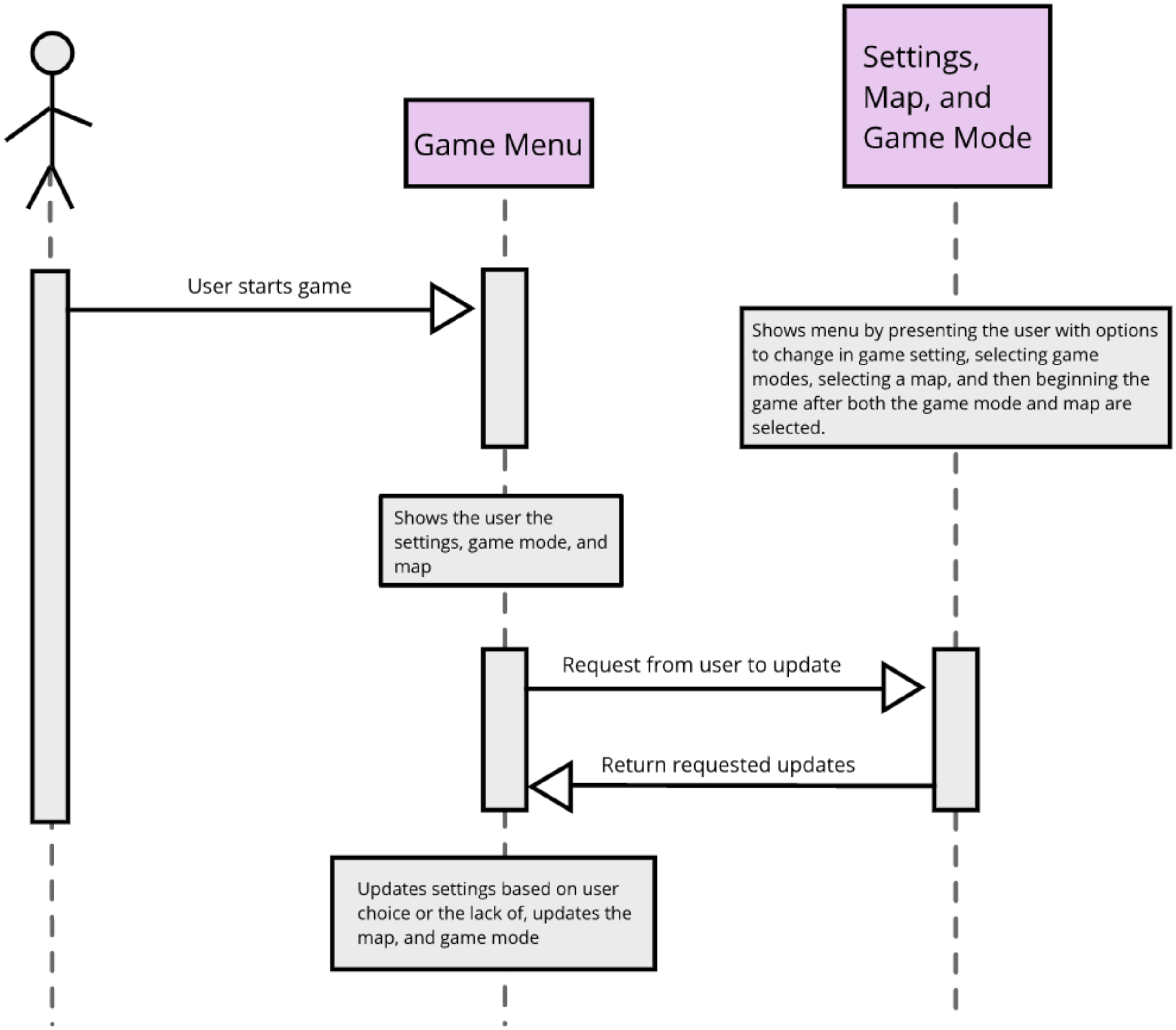
Range Only

This game mode is similar to Melee Only, except instead of being limited to melee attacks, the players are limited to using ranged weapons. This causes a game style that encourages players to make make well timed and well aimed projectile attacks, similar to an airsoft game in real life but with slo mo effects.

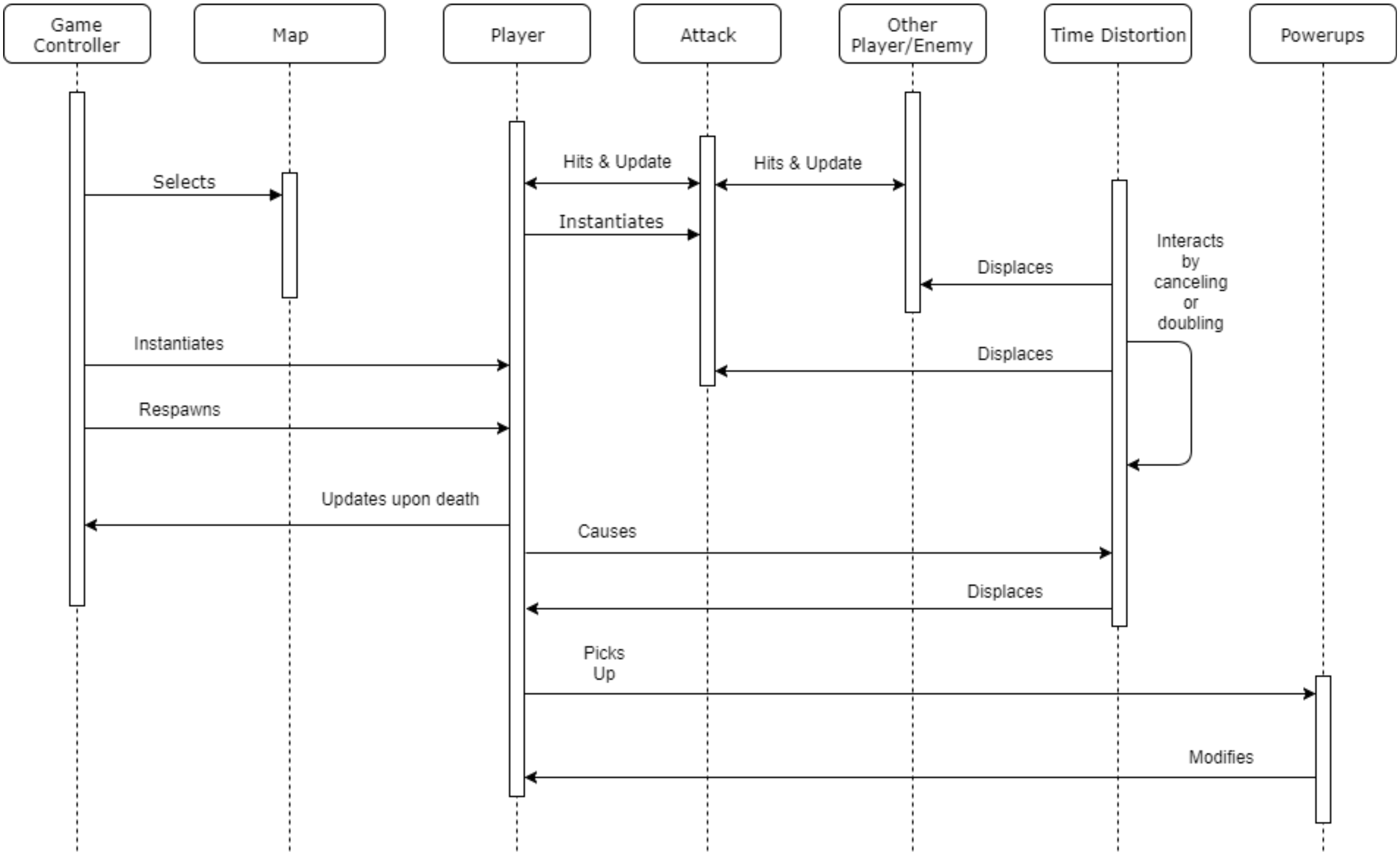
Exponential Mode

In this game mode, anytime a player is hit by an attack, their time-gauge beings to deplete itself at a faster rate. Each successive hit makes their gauge drop faster and faster, stacking exponentially until the player dies. This will encourage a game mode that is less frantic and more thought out, as each player will attempt to poke each other. Initially, all players will have a much slower than usual time-gauge drop rate, helping to balance against a game mode that intentionally increases depletion rate of the time-gauge.

Sequence of Events When User First Starts Game from the menu:

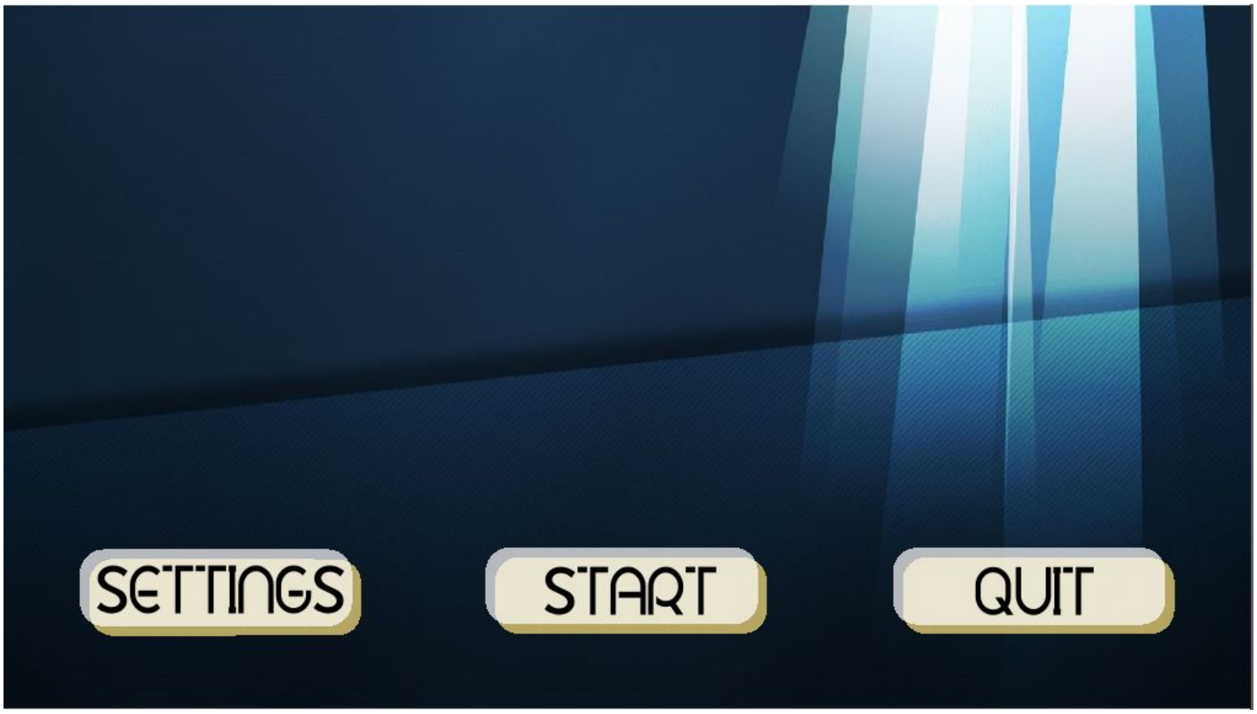

Settings, Map, and Game Mode
Game Menu
User starts game
Shows menu by presenting the user with options to change in game setting, selecting game modes, selecting a map, and then beginning the game after both the game mode and map are selected.
Shows the user the settings, game mode, and map
Request from user to update
Return requested updates
Updates settings based on user choice or the lack of, updates the map, and game mode

Sequence of Events when Player plays the game:

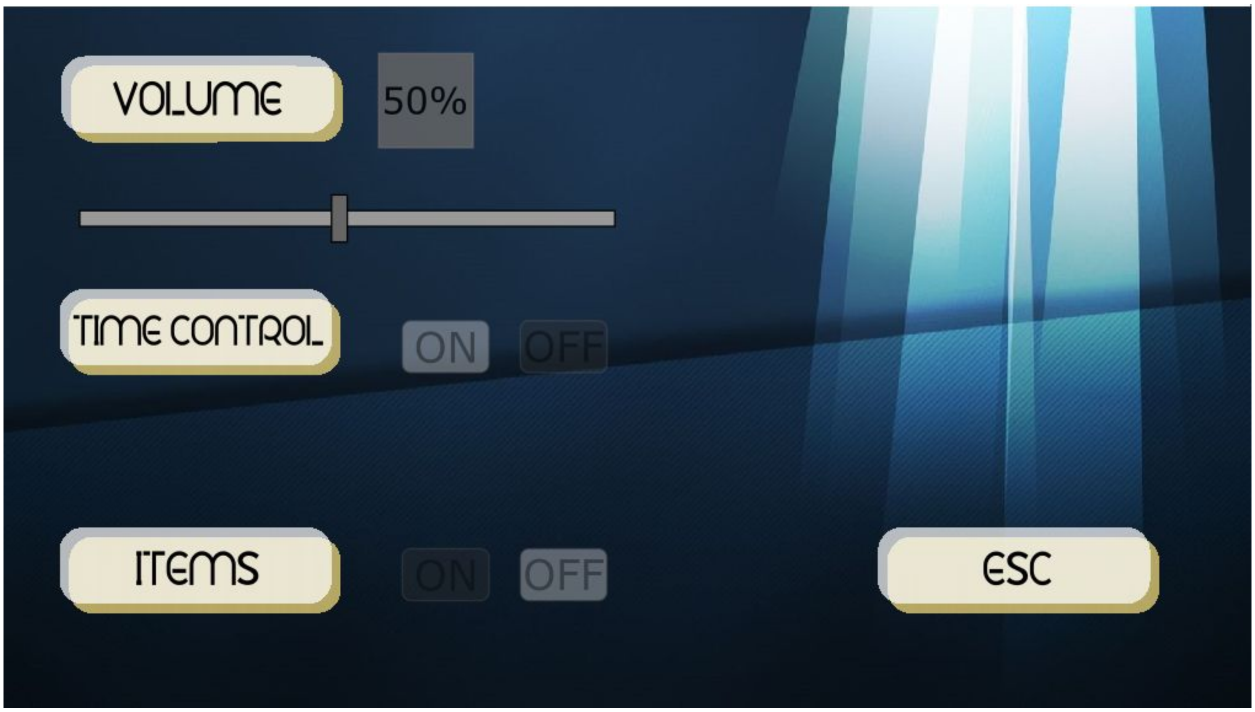


Menu Mockups:

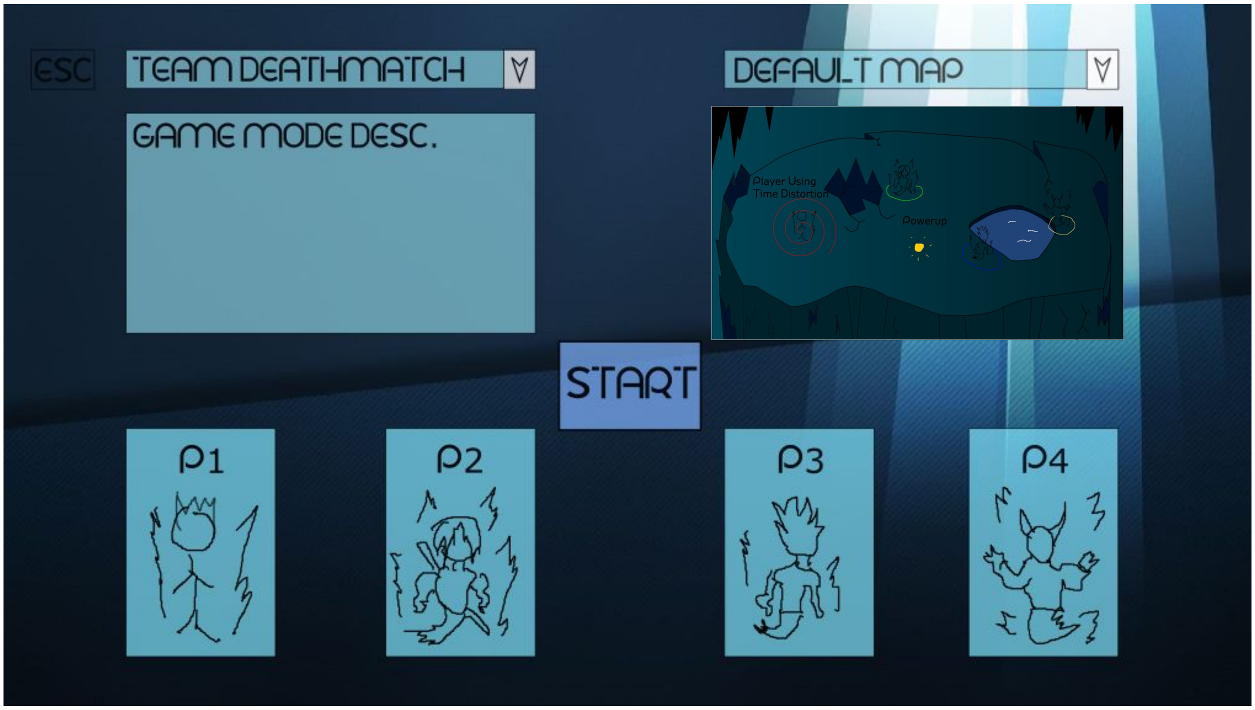
Main Menu Mockup:



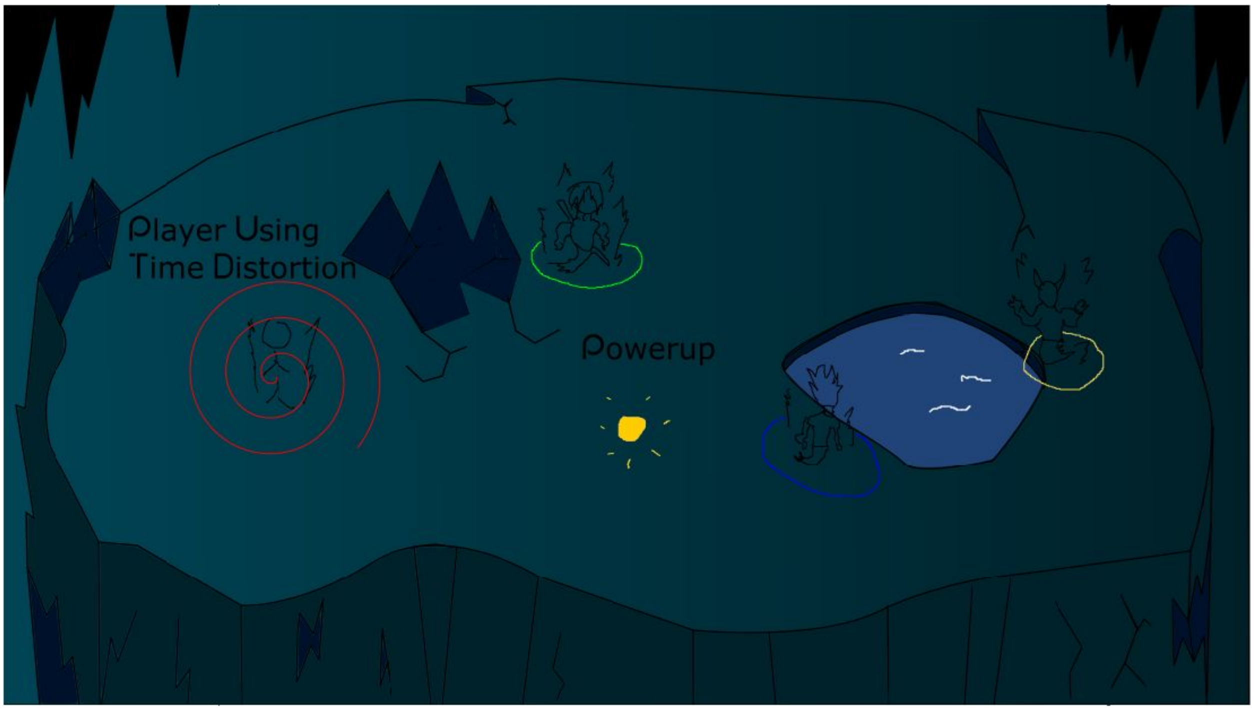
Settings Menu Mockup:



Game Setup Mockup:



Gameplay:


Player Using
Time Distortion
Powerup