***Coliseum - Technical Specifications***

Jun Yen Leung - junyenle@usc.edu

Ethan Barker - esbarker@usc.edu

Jake Leventhal - jglevent@usc.edu

Uri Rahimi - urahimi@usc.edu

Dylan Appleton - dappleto@usc.edu

**Section 0: Misc Notes**

* Platform: Coliseum will run on Windows Operating Systems and will be designed on Windows 10.
* Coliseum will use keyboard controls.

**Section 1: Title Screen, Login, Registration**

* Title Screen / Login / Registration (8 Hours)
  + The main title screen needs to have four buttons: login, register, view s, and continue as guest.
    - If the login button is clicked, the user needs to be brought to a screen with a form that takes two text fields: username and password. Here, there must be a back button which returns the user to the previous screen, and a login button that submits the form. Clicking this login button should check the username and password inputs against the database. If the username and password are valid, the user will be logged in. If not, an error message should be displayed.
    - If the register button is clicked, the user needs to be brought to a screen that has a username field, password field, confirm password field, register button, and back button. The back button should send the user back to the previous screen. Clicking the register button will only allow the user to register if all of the following conditions are met:
      * All fields have entries.
      * The username field does not match any usernames already in the database.
      * Both password fields contain the same text.
      * “Registering” will make an entry to a database storing this new user’s details and return the user to the main screen.
    - If the “continue as guest” button is clicked, we will let the user proceed and to the main game as a “guest user” that is not able to save their score (but otherwise able to make use of all game features).
    - The view s button will display the list of top ten s achieved worldwide.

**Section 2: Agents**

**2a: Player Character (PC)**

PC Actions (8 Hours)

* + The PC should have the following abilities within the game world:
    - If the spacebar is pressed, the PC should jump. Holding down the spacebar should result in a higher jump.
    - If the ‘a’ button is pressed, the PC should move to the left.
    - If the ‘d’ button is pressed, the PC should move to the right.
    - If the ‘j’ button is pressed, the PC should attack forwards.
    - Holding down ‘a’ or ‘d’ alongside ‘j’ should cause the PC to attack in the specified direction.
    - Holding down shift while moving should cause the PC to dash.
    - Attacking while in the air should cause the PC to perform a forward attack mid-air.
    - The PC’s attack range should be short and based on the length of the PC’s sword (see PC Art/Animations below).
    - The PC should have a “hitbox”. This hitbox is a rectangle around the PC’s sprite (see PC Art/Animations below) that determines when the PC gets hit by hostile elements. The hitbox should be set frame-by-frame for each of the PC’s animations.
    - If the PC is hit, it should lose one health point (out of a maximum of three). These health points should be visible to the player.
    - If the PC runs out of health points, it should die.

PC Art/Animations (8 Hours)

* + The PC should be drawn as a 200(w)x400(h) pixel vector drawing.
  + The PC should have 6-12 frame-per-second animations for all of its actions as well as a flinch animation and a death animation. The exact frames per second will depend on the complexity of the animation itself, but it must always be a multiple of 6.
  + All animations should be accompanied by an appropriate sound effect.

PC Design Sketch ^

**2b: Hostile Agent (BOSS)**

BOSS Actions (8 Hours)

* + The BOSS should have the following abilities within the game world:
    - The BOSS should be able to move and attack (exactly how depends on the design of the BOSS, but these two broad abilities are essential).
    - The BOSS will pick actions using a decision tree that takes into account factors such as PC position and distance.
    - The BOSS should have a “hitbox”. This hitbox is a rectangle around the BOSS’ sprite (see BOSS Art/Animations below) that determines when the BOSS gets hit by the PC’s attacks. The hitbox should be set frame-by-frame for each of the BOSS’ animations.
    - If the BOSS is hit, it should lose one health point (maximum health to be determined). These health points should be invisible to the player.
    - If the BOSS runs out of health points, it should die.

BOSS Art/Animations (8 Hours)

* + The BOSS should be drawn as a vector drawing. The size of the drawing depends on the design of the BOSS.
  + The BOSS should have 6-12 frame-per-second animations for all of its actions as well as a flinch animation and a death animation. The exact frames per second will depend on the complexity of the animation itself, but it must always be a multiple of 6.
  + The BOSS should have 6-12 frame-per-second flare animations (for example, roaring or cracking its knuckles) to make it seem more life-like.
  + All animations should be accompanied by an appropriate sound effect. Boss Concept Sketch ^

Additional BOSS Notes:

* + There should be one BOSS per level.
  + Each BOSS should have its own music track. This track should fit the “personality” of the BOSS.
  + BOSS actions should be clearly telegraphed so that the player can anticipate what the BOSS is about to do.

**Section 3: Environment**

**3a: Player-acting environment**

Environment Design (8 Hours)

* + By player-acting, we mean environment features that the PC can interact with. This will be limited to platforms and drops.
  + From here onwards, the environment in which the PC and BOSS interact shall be referred to as the LEVEL.
  + The platform will be a 4000(w)x200(h) pixel sprite.
  + This sprite will be positioned and given a collision box in Unity.
  + The PC and any BOSS should be able to stand on the platform and move along it.
  + Moving off the platform should result in the death of the PC.
  + We will simulate gravity by pulling the PC and BOSS downwards (that is, manipulating their y position) whenever they are in the air.
    - Combining the last two points: moving off the platform should result in the PC falling to the bottom of the screen and dying instantly.
  + The LEVEL should be pre-created and stored on the server. Beginning starting a new game should load the LEVEL.

**3b: Non-player-acting environment**

Environment Design (8 Hours)

* + By non-player-acting, we mean environment features that are purely aesthetic.
  + The LEVEL should, in addition to its player-acting features, have a background, midground, and foreground.
  + The platform should exist behind the foreground but in front of the midground. The background should exist in the very back.
  + As the PC moves across the screen, the background, midground, and foreground should move in the opposite direction. The background should move the least and the foreground the most. This is known as parallax motion and is done by rendering the background, midground and foreground art differently depending on its “parallax factor”. This parallax factor will be tracked as a float variable.
    - This allows us to create the illusion of depth with only 2D art.
  + The background, midground, and foreground should also include small looping animations such as moving clouds or insects in the grass. This makes the game seem more life-like.
    - This will be done by creating separate threads for said animations and having them loop indefinitely.

**3c: Player / Environment Bridging**

By this we mean the elements that help connect the player on the other side of the screen to the environment we have created.

LEVEL Loading (1 Hour)

* + When a new game begins, the screen should fade out to black. It should fade back in with the LEVEL already loaded and the PC/BOSS already positioned at their starting positions on the LEVEL.

Player Camera (8 Hours)

* + The Player Camera (PLAYCAM) determines which elements of the LEVEL are visible on the player’s screen.
  + The PLAYCAM should be centered on the PC and move with the PC.
  + The width of the PLAYCAM should be the width of the player’s screen, up to a maximum of 1920 pixels.
  + This will be done by updating the x and y positions of the PLAYCAM’s center every few frames and setting it equal to the PC’s x and y positions.
    - We do this every few frames and not every frame so that we can achieve a delayed camera following effect (for style purposes).

**3d: LEVEL Concept Sketch**

****

**Section 4: Additional features**

**4a: Highscores**

Saving Highscores (3 Hours)

* + If the player is both a registered user and beats the BOSS, their score will be calculated based on how much time they spent and whether they were able to beat the BOSS without taking damage.
  + This score will be sent to the server, which will check if the score against the highscore database to determine if it is one of the top 10 scores across all saved player scores.
    - Score will be calculated by taking the quotient of the quickest time the BOSS can possibly be defeated in and the time that the player took to defeat the BOSS and multiplying said quotient by 10,000. 5,000 bonus points should be awarded for defeating the BOSS without taking any damage.
    - For example, on a BOSS that can at best take 5 minutes, a player who beats it in 10 minutes without taking damage will score 5/10 \* 10,000 + 5,000 = 10,000 points.
  + If so, the highscore database will be updated to reflect this new highscore.
  + Players will, regardless of whether their new score is a highscore, be shown a screen displaying their personal score alongside the highscore list. If the player’s score is a new highscore, it should be displayed in a different color so that it is easily visible.
  + Players playing as guests will *not* be able to save their scores, but will still be shown the highscore list at the end.

Viewing Highscores (1 Hour)

* + On the post-login screen (see above), if the user chooses to access the highscores list, the game will query the highscore database and display the top 10 scores achieved across all players to the user.

**4b: GAME OVER**

GAME OVER Event (2 Hours)

* + GAME OVER can occur in two ways:
    - The PC dies: a GAME OVER screen should be displayed and the screen will fade to black. The screen will fade back to the title screen.
    - The BOSS dies, in which case the word “VICTORY”, the player’s score , and the highscore list should be displayed as specified in section 4a. Afterwards, the screen will fade back to the title screen.

**Section 5: Database**

Database (8 Hours)

* + We will use a MongoDB database hosted on one of the project member’s desktop computer.
  + The database (DB) will have two collections:
    - The first collection will consist of all registered players and their login details. For instance,

{

username : “ttrojan”,

password : “fighton”

}

This collection should be indexed by username for quick querying when players try to log in.

* + - The second collection will consist of the top 10 highscores and the usernames of the players that achieved them. For instance,

{

username : “ttrojan”,

score: “5000”

},

{

username : “bbruin”,

score: “4000”

}