Team: Alpha-Bravo

Members: Anthony Martinez (Team Leader), Dustin Chhum, Howard Cho, Luis Rodriguez

**Requirements Analysis Document**

1. **Introduction** 
   1. **Purpose**:The client, SNK Playmore, is a large-scale publishing company who is looking to bolster its existing fan base. To achieve this they want to us to incorporate a web-based game to their existing website. The game should be a platformer that can be deployed with cross-browser compatibility. It should also allow customers to play for more than just a few minutes, thus options to save and retrieve games should be included.

* 1. **Goal**: The game should allow current customers to start new games, continue saved games, and track their scores versus other players. This will require the development of a game that is specifically deployable to web browsers and interacts with SNK’s user database. Merging the game with each user’s profile will create a richer experience for each user.
  2. **Scope of system**:
     1. The game should provide functionality that allows users to start new games, load saved games, and view leaderboards. These options will be accessed using a GUI from a keyboard accessible computer.
     2. The game should include enemies that have AI.
     3. The game should include weapons, coins, and health items.
     4. The game should include a game map that is predefined.
     5. The game will include a small database that will only save game data based on the user’s id.
     6. The frame rate per second will be at least 30.
     7. The game will be deployable on the latest version of Firefox, IE, and Chrome only.
     8. Social media sharing options will not be included.
     9. A game store will not be included.
     10. Micro purchasing will not be included.
     11. Multiplayer capabilities will not be included.
     12. A new website will not be included.
     13. Customizable game control options will not be included.
  3. **Objectives and fulfillment criteria**: The following functional and nonfunctional requirements will be met:
     1. Start/Save/Load a game.
     2. View the leaderboards.
     3. Play a game.
     4. Fight enemies.
     5. Navigate through obstacles.
     6. Obtain weapon, coin, and health items.
  4. **Synopsis:** The following are detailed descriptions of the proposed game’s functional and nonfunctional requirements. Also included are some scenarios of possible user interaction with the game, use cases, and finally game mock-ups.

1. **Current System**: No current system exist.
2. **Proposed Game:**
   1. **Overview**
      1. The game will allow the player to use a keyboard for input and move the character around in the level.
      2. The player will be able to interact with enemies, obstacles, and various items.
      3. The game should have the capability of saving the player's data and progress.
      4. The game should have a leaderboard, so you can see how you are doing versus other players.
      5. The game should allow the player to attack an enemy using weapons found throughout the level.
   2. **Functional Requirements**
      1. Using the Main Menu
         1. Player must already be logged in the website to use the main menu.
         2. Website establishes a connection to the player’s id and pulls their data from the server and downloads it to the player’s screen.
         3. A menu will display the following on opening:
3. New Game
4. Load Game
5. Leaderboards
6. Exit
   * 1. Starting a Game
        1. Player must in the main menu.
        2. Player must navigate to the option to enter a new game.
        3. The game will load the following assets:
7. Map (graphical assets)
8. Player model
9. Enemy models
10. Game audio
11. Game variables
    1. Health
    2. Level
    3. Inventory
    4. Score
       * 1. The server will save the game to the database and associate it to the Player’s id.
       1. Viewing Leaderboards
          1. Player must already be logged into the website.
          2. Player must navigate to the leaderboards screen.
          3. The server’s database will retrieve the player’s leaderboard data and display it to the player.
          4. The leaderboards will display the following on opening:
12. [User ID] [Level 1] [Ranking 1]
13. [User ID] [Level 2] [Ranking 2]
14. [User ID] [Level 3] [Ranking 3]
    * 1. Save game
         1. The user must navigate to the settings menu. Accessing this menu will pause the current game state.
         2. The settings menu will display the following on opening:
15. Save Game
16. Exit Game
    * + 1. When the save option is selected the current state of the game is saved to the game database. Data saved includes:
17. Health
18. Level
19. Inventory
20. Score
21. Game objects
22. Player location
    * 1. Play a game
         1. A game will commence once all game assets are loaded.
         2. The game session will load and manage following data fields:
23. Health: A percent between 1 and 100
24. Level: Numeric
25. Inventory A list of items
26. Score Numeric
    * + 1. The game session will also map user controls to the keyboard.
27. Move right: Keyboard key
28. Move left: Keyboard key
29. Jump: Keyboard key
30. Attack: Keyboard key
31. Settings: Keyboard key
    * 1. Encounter an enemy
         1. Enemies should spawn at pre-planned locations on the map.
         2. When the player comes within a pre-defined distance of the enemy, the enemy’s AI should commence its attack tactics.
         3. The player will be able to switch weapons to attack and defend against enemies. The following will allow for the player to change weapons:
32. Weapon 1: Number key
33. Weapon 2: Number key
34. Weapon 3: Number key
35. Weapon 4: Number key
36. Weapon 5: Number key
    * + 1. The enemy displays a health meter above itself.
37. Health: Bar meter
    * + 1. The resistance of the enemy to the player’s weapons will be determined by the level of the enemy.
38. Resistance on easy Player’s weapon fully effective
39. Resistance medium Player’s weapon 80% effective
40. Resistance hard Player’s weapon 60% effective
41. Resistance very hard Player’s weapon 45% effective
    * 1. Exit the Game
         1. The player selects the settings menu on the top right corner.
         2. This causes the game state to pause.
         3. The settings menu will consist of the following:
42. Exit the game
43. Save the game
    * + 1. Once a player selects exit the game a prompt will appear asking for verification.
        2. Then the game exits.
      1. Encounter a Gap and Misses
         1. The player encounters a gap at a predetermined location.
         2. The gap will contain spikes that will cause the player to lose the game.
         3. The player uses the jump key to jump over the gap but misses and gets impaled by the spikes.
         4. Then the players health automatically goes to zero, the following updates on the Heads Up Display (HUD):
44. Health: Bar Meter to Zero
    * + 1. The following data gets save for tabulating the score for the leaderboard:
45. UserID
46. Score
    * + 1. Then the game ends and displays the following:
47. Level
48. Score
    * 1. Encounter a Gap and Makes It
         1. The player encounters a gap at a predetermined location.
         2. The gap will contain spikes that will cause the player to lose the game.
         3. The player uses the jump key to jump over the gap and makes it.
         4. Then the player makes it and continues on with the level.
      2. Encounter a Platform
         1. The player encounters a platform at a predetermined location.
         2. The platform itself will be elevated high enough so the player cannot just jump onto.
         3. The player must find another object to maneuver onto the elevated platform.
         4. Then once the player justs said object and uses it to maneuver onto the elevated platform, they continue on with the game.
      3. Encounter a pick-up item and avoid it
         1. While the player progresses through the level, the player encounters a pick-up item which will have both a predetermined spawn location and item type.
         2. The player walks past the item and does not enter key to pick it up.
         3. The player continues with the current level.
      4. Encounter a pick-up item and picks it up
         1. While the player progresses through the level, the player encounters a pick-up item which will have both a predetermined spawn location and item type.
         2. The player walks up to the item, then presses a key to pick it up.
         3. The item type will determine what stats get updated

1. Item Inventory Bar

2. Score

* + - 1. The player continues through the level.
    1. Encounter an auto pick-up item and avoids it
       1. While the player progresses through the level, the player encounters an auto pick-up item which will have both a predetermined spawn location and item type.
       2. The player will jump over or takes a different path and does not make contact with the item.
       3. The player continues through the level.
    2. Encounter an auto pick-up item and makes contact it
       1. While the player progresses through the level, the player encounters an auto pick-up item which will have both a predetermined spawn location and item type.
       2. The player walk up to the item and make contact with it.
       3. Once contact was made it automatically gets picked up.
       4. The item type will determine what stats get updated.

1. Item Inventory Bar

2. Score

3. Health

* + - 1. The player continues through the level.
  1. **Non-Functional Requirement**s

User interface

1. The user interface must be user-friendly.
2. The menu should load within 5 seconds.

Performance

1. The game should have a stable game frame rate of 30 frames per second.
2. The game should have a response time of .1 seconds for click operations.

Hardware supported

1. A desktop or laptop computer with an integrated graphics chip is required.

I/O devices

1. A working mouse,keyboard, and monitor is required.

Enemies

1. Enemies must have AI that actively attacks the player.

Items

1. Items must stay on screen until player leaves the current frame.
2. All items must have an equal chance to appear.
3. Items obtained must be responsive within .1 seconds.
4. Game statistics should update within .1 seconds upon player interaction with enemies or items that cause health, score, or inventory changes.

Error Handling

1. The game should handle invalid inputs.
2. In the event the game crashes, it should restart within a 3 minute time frame.
3. The game should handle errors caused by loss of internet connections.

Documentation

1. A user guide will be provided on how to play the game.
2. A document for users to see if system is compatible with the minimum specs required by the game.
3. Database documentation will be provided.

Security

1. The game database will be turned over to key personnel.
2. User information passed from the website to the game database will be encrypted.

Web Browser

1. The game will include WebGL support.
2. The game will be compatible with the latest version of Firefox, IE, and Chrome.
   1. **Scenarios**

Actors: Player

Players: The player opens the program, starts the game, and controls the main character.

1. Start Screen

|  |  |
| --- | --- |
| Scenario Name: | 1a). Start Screen |
| Actors: | Player |
| Flow of Control: | 1. The player selects “New Game” from the GUI. 2. The screen displays “loading...”. 3. The first level is displayed along with the following information:   Health meter: 100  Level: 1  Player Inventory Empty  Score 0 |

|  |  |
| --- | --- |
| Scenario Name: | 1b). Start Screen |
| Actors: | Player |
| Flow of Control: | 1. Player selects “Leaderboards” from the GUI. 2. The screen displays the best scores. 3. The player selects “Back to Main Menu”. 4. Go back to Start Screen. |

|  |  |
| --- | --- |
| Scenario Name: | 1c). Start Screen |
| Actors: | Player |
| Flow of Control: | 1. Player selects “Exit” from the GUI. 2. The game displays an exit screen showing the company logo. 3. The game exits to the desktop. |

2. Play a game

|  |  |
| --- | --- |
| Scenario Name: | 2a). Play a game |
| Actors: | Player |
| Flow of Control: | 1. Player selects “Load Game” from the GUI. 2. The screen displays the list of saved games. 3. The player selects a game from the list.   Saved Games:  ..Tony saved on 1 9/01/2016 (game one )  ..Tony saved on 19/02/2016 (game two)   1. Game two is selected and loaded.   Health meter: 87  Level: 1  Player Inventory Empty  Score 1516 |
|  | 1. The player proceeds through the level by moving towards the right and encounters a level 2 tank enemy. 2. The enemy attacks the player with a gun attack. 3. The player avoids the attack and jumps over the tank. 4. The player continues with the level. |

|  |  |
| --- | --- |
| Scenario Name: | 2b). Play a game |
| Actors: | Player |
| Flow of Control: | 1. The player selects “Load Game” from the GUI. 2. The screen displays the list of saved games. 3. The player selects a game from the list.   Saved Games:  ..Tony saved on 1 9/01/2016 (game one)  ..Tony saved on 19/02/2016 (game two)   1. Game one is loaded and the current level is displayed along with saved data.   Health meter: 67  Level: 1  Player Inventory Gun <current weapon>  Ground To Air Rocket  Score 78524 |
|  | 1. The player proceeds through the level by moving towards the right and encounters a Mark I flying robot. 2. The enemy attacks the player with a missile attack. 3. The player is struck by the attack and loses 30 health points. 4. The health meter now reads 37. 5. The player opens the inventory menu that shows:     Inventory List:  Gun <current weapon>  Ground To Air Rocket   1. <Empty> 2. <Empty> 3. <Empty> |
|  | 1. The player selects Ground To Air Rocket. 2. The new weapon is equipped and the player fires at the Mark I. 3. The enemy is struck and destroyed. 4. The player continues with the level. |
|  |  |

|  |  |
| --- | --- |
| Scenario Name: | 2c). Play a game |
| Actors: | Player |
| Flow of Control: | 1. Player selects “New Game” from the GUI. 2. The screen displays “loading...”. 3. The first level is displayed along with the following information:   Health meter: 100  Level: 1  Player Inventory Empty  Score 0 |
|  | 1. The player proceeds through the level by moving towards the right and encounters a pallet of barrels. 2. The player jumps over the pallet of barrels and continues with the level. |
|  |  |
|  |  |

3. Encounter an obstacle

|  |  |
| --- | --- |
| Scenario Name: | 1a). Encounters an obstacle |
| Actors: | Player |
| Flow of Control: | 1. Player encounters a spike pit of death. 2. Player tries to jump over spike pit of death and fails. 3. Player falls down the pit and gets impaled by spikes and dies. 4. The game is over. |

|  |  |
| --- | --- |
| Scenario Name: | 1b). Encounters an obstacle |
| Actors: | Player |
| Flow of Control: | 1. Player encounters a wall too high to jump over. 2. The player uses a nearby stack of boxes to jump on to. 3. The player then jumps from the boxes to the elevated platform. 4. The player continues with the level |

4. Encounter a weapon

|  |  |
| --- | --- |
| Scenario Name: | 1a). Encounter a weapon |
| Actors: | Player |
| Flow of Control: | 1. As the player is moving along the level he comes across a machine gun that can be added to his inventory. 2. The player positions himself next to the machine gun and chooses to pick it up. 3. The inventory is adjusted to reflect the addition of the new weapon.     Inventory List:  Machine gun  Gun <current weapon>  Ground To Air Rocket  <Empty>  <Empty> |
|  | 4. The player selects the machine gun while the inventory list is still opened and equips it.  5. The inventory list is updated    Inventory List:  Machine gun <current weapon>  Gun  Ground To Air Rocket  <Empty>  <Empty>  6. The player continues with the level. |

|  |  |
| --- | --- |
| Scenario Name: | 1a). Encounter a weapon |
| Actors: | Player |
| Flow of Control: | 1. As the player is moving along the level he comes across a ray beam gun that can be added to his inventory. 2. The player positions himself next to the ray beam gun and chooses to pick it up. 3. The inventory is adjusted to reflect the addition of the new weapon.     Inventory List:  Ray beam gun  Gun <current weapon>  Ground To Air Rocket  <Empty>  <Empty> |
|  | 1. The player closes the inventory list and continues with the level. |

5. Encounter non-weapon item

|  |  |
| --- | --- |
| Scenario Name: | 1a). Encounter non-weapon item |
| Actors: | Player |
| Flow of Control: | 1. The player encounters a treasure chest. 2. The player positions himself next to the treasure chest and chooses to open it. 3. The treasure chest opens up. 4. Player receives all loot in chest. 5. Player continues with the level. |

|  |  |
| --- | --- |
| Scenario Name: | 1b). Encounter non-weapon item |
| Actors: | Player |
| Flow of Control: | 1. Player encounters a 50 point health power-up. 2. The player walks through the power-up and receives 50 points of health. 3. Player continues with the level. |

|  |  |
| --- | --- |
| Scenario Name: | 1c). Encounter non-weapon item |
| Actors: | Player |
| Flow of Control: | 1. The player encounter a 100 point coin. 2. The player walks through the power-up and receives 100 points to the score. 3. The player continues with the level. |

* 1. **Use Cases**

|  |  |
| --- | --- |
| Use-Case Name: | Exiting the Game |
| Actors: | Player |
| Pre-Conditions | 1. Player must be in the main menu. |
| Flow of Control | 1. Player enters the option to exit the game.    1. An exit screen appears. |
| Post Conditions | 1. Player is now no longer in the main menu. |
| Error Conditions | 1. Data doesn’t save upon exit. |
| Non-Functional Requirements | 1. The player’s data needs to be saved into the database upon exit. |

|  |  |
| --- | --- |
| Use-Case Name: | In-Game Settings |
| Actors: | Player |
| Pre-Conditions | 1. Player must be in-game. |
| Flow of Control | 1. Player presses button on the top right of the screen. 2. A menu shows up showing a variety of options.    1. See Use Case: Exit a Game (In-Game)    2. See Use Case: Save a Game (In-Game) |
| Post Conditions | 1. Player is in the in-game settings screen. |
| Error Conditions | 1. Game hangs after clicking the settings button. |
| Non-Functional Requirements | 1. Menu should pop up instantaneously. |

|  |  |
| --- | --- |
| Use-Case Name: | Save a Game (In-Game) |
| Actors: | Player |
| Pre-Conditions | 1. Player must be in the settings menu. |
| Flow of Control | 1. Player selects option to save game. 2. The game is saved to the database. |
| Post Conditions | 1. The player’s game is saved to the database. |
| Error Conditions | 1. The player’s game isn’t saved to the database. |
| Non-Functional Requirements | 1. The game must saved in less than 1 second. |

|  |  |
| --- | --- |
| Use-Case Name: | Exit the Game (In-Game) |
| Actors: | Player |
| Pre-Conditions | 1. Player must be in settings menu. |
| Flow of Control | 1. Player selects the option to exit the game. 2. The players data is saved into the database. 3. A prompt is shown telling the Player their data and game is being saved. 4. The game exits the level. 5. See Use-Case: Using the Main Menu |
| Post Conditions | 1. Player exits to the Main Menu. |
| Error Conditions | 1. Player’s data isn’t saved upon exit. 2. Database connection fails. 3. The prompt is not shown when the data is saved. 4. The game hangs after the player selects the exit option. |
| Non-Functional Requirements | 1. Game should exit to the main menu in less than 2 seconds. |

|  |  |
| --- | --- |
| Use-case Name: | Play a Game |
| Actors: | Player |
| Pre-Conditions: | 1. User must be on the website. 2. User must be logged on. 3. User must be using a keyboard. 4. User must be at the game menu. |
| Flow of Control: | 1. The player selects a game type from a drop box    1. [ game type] 2. The game type is loaded and displays the following data:    1. [health]    2. [level]    3. [inventory]    4. [score] 3. The player proceeds through the level by moving towards the goal. |
| Post-Conditions: | 1. A game and its data is loaded. 2. Game data can be saved. |
| Error-Conditions: | 1. Game type selected not available. 2. Game hangs when loading and never loads. 3. Web browser can not load the game. 4. Frames per second drops below 30 FPS. |
| Non-Functional Requirements: | 1. Game loads within 5 seconds. 2. Cross browser compatible. 3. WebGL support included. 4. The game must run at between 30- 60 frames per second. |

|  |  |
| --- | --- |
| Use-case Name: | Encounter and Avoid an Enemy |
| Actors: | Player |
| Pre-Conditions: | 1. User must have loaded a game. 2. A an enemy must have spawned. |
| Flow of Control: | 1. The player encounters an enemy. 2. The player avoids enemy attacks and maneuvers past it. 3. The player continues with the level. |
| Post-Conditions: | 1. The current game is still ongoing. 2. The player statistics do not change. |
| Error-Conditions: | 1. Controls are not responsive. 2. Controls lag. |
| Non-Functional Requirements: | 1. Controls must be responsive within .1 seconds 2. The ground enemy must actively attack the player. |

|  |  |
| --- | --- |
| Use-case Name: | Encounter and Engage an Enemy |
| Actors: | Player |
| Pre-Conditions: | 1. The user must have loaded a game. 2. An enemy must have spawned. 3. The player does not have a weapon selected. |
| Flow of Control: | 1. The player encounters a ground enemy. 2. The enemy attacks the player. 3. The player is struck and loses health. 4. The player’s following statistics change:    1. [health] 5. The player opens the inventory menu    1. [weapon item]    2. [weapon item]    3. [weapon item] 6. The player selects one of the weapon items. 7. The player equips the item and fires at the enemy. 8. The player’s following statistics change:    1. [score] 9. The enemy is struck an amount of times and dies. 10. The player continues with the level. |
| Post-Conditions: | 1. The current game is still ongoing. 2. The player’s health level changes. 3. The player’s score changes. |
| Error-Conditions: | 1. Controls are not responsive. 2. Controls lag. 3. Weapon does not load onto the character. |
| Non-Functional Requirements: | 1. Controls must be responsive within .1 seconds. 2. The ground enemy must actively attack the player. |

|  |  |
| --- | --- |
| Use-case Name: | Encounter a Gap and Misses |
| Actors: | Player |
| Pre-Conditions: | 1. User must have loaded a game. 2. The obstacle must have appeared. |
| Flow of Control: | 1. The player encounters a gap. 2. The player tries to jump the gap. 3. The player misses the jump. 4. The player loses an amount of health. 5. Adjusts the following data:    1. [health] |
| Post-Conditions: | 1. The current game is on going. 2. The player’s health level changes. |
| Error-Conditions: | 1. Controls are not responsive/lag. 2. The gap does not appear. |
| Non-Functional Requirements: | 1. Controls must be responsive within .1 seconds. |

|  |  |
| --- | --- |
| Use-case Name: | Encounter a Gap and Makes It |
| Actors: | Player |
| Pre-Conditions: | 1. User must have loaded a game. 2. The obstacle must have appeared. |
| Flow of Control: | 1. The player encounters an obstacle. 2. The player tries to jump the obstacle. 3. The player makes the jump. 4. The player continues on with the level. |
| Post-Conditions: | 1. The current game is still ongoing. 2. The player statistics do not change. |
| Error-Conditions: | 1. Controls are not responsive/lag. 2. The gap does not appear. |
| Non-Functional Requirements: | 1. Controls must be responsive within .1 seconds. |

|  |  |
| --- | --- |
| Use-case Name: | Encounter a Platform |
| Actors: | Player |
| Pre-Conditions: | 1. User must have loaded a game. 2. The obstacle must have appeared. |
| Flow of Control: | 1. The player encounters a platform that is elevated. 2. The player uses other objects to maneuver onto the platform. 3. The player continues with the level. |
| Post-Conditions: | 1. The current game is still ongoing. 2. The player statistics do not change. |
| Error-Conditions: | 1. Controls are not responsive/lag. 2. Nearby object does not load. |
| Non-Functional Requirements: | 1. Controls must be responsive within .1 seconds. |

|  |  |
| --- | --- |
| Use-case Name: | Encounter and Avoid a Pick-up Item |
| Actors: | Player |
| Pre-Conditions: | 1. User must have loaded a game. 2. An item must be spawned. |
| Flow of Control: | 1. The player encounters a pick up item. 2. The player avoids the item and moves passed it. 3. The player continues with the level. |
| Post-Conditions: | 1. The current game continues. 2. The player statistics do not change. |
| Error-Conditions: | 1. Controls are not responsive. 2. Controls lag. 3. Item does not load. |
| Non-Functional Requirements: | 1. Controls must be responsive within .1 seconds 2. Item must remain on-screen until next screen frame. 3. Item must have an equal chance of spawning. |

|  |  |
| --- | --- |
| Use-case Name: | Encounter and picks up pick-up item |
| Actors: | Player |
| Pre-Conditions: | 1. User must have loaded a game. 2. An item must be spawned. |
| Flow of Control: | 1. The player encounters a pick-up item. 2. The player click a button and picks up item. 3. The player gains item picked up. 4. Adjust statistics to player inventory or score 5. [ Item] 6. [Score] 7. The player continues with the level. |
| Post-Conditions: | 1. The current game continues. 2. The player statistics changes. 3. The player’s inventory changes. |
| Error-Conditions: | 1. Controls are not responsive. 2. Controls lag. 3. Item does not load.   4. Item does not register being picked up. |
| Non-Functional Requirements: | 1. Controls must be responsive within .1 seconds. 2. Item must remain on-screen until next screen frame. 3. Item must have an equal chance of spawning. 4. Stats from item gain must update within .1 seconds |

|  |  |
| --- | --- |
| Use-case Name: | Encounter and avoid an auto-pick up item |
| Actors: | Player |
| Pre-Conditions: | 1. User must have loaded a game. 2. An item must be spawned. |
| Flow of Control: | 1. The player encounters an auto-pick up item. 2. The player avoids the item and moves passed it. 3. The player continues with the level. |
| Post-Conditions: | 1. The current game continues. 2. The player statistics do not change. |
| Error-Conditions: | 1. Controls are not responsive. 2. Controls lag. 3. Item does not load. |
| Non-Functional Requirements: | 1. Controls must be responsive within .1 seconds. 2. Item must remain on-screen until next screen frame. 3. Item must have an equal chance of spawning. |

|  |  |
| --- | --- |
| Use-case Name: | Encounter and gets an auto-pick up item |
| Actors: | Player |
| Pre-Conditions: | 1. User must have loaded a game. 2. An item must be spawned. |
| Flow of Control: | 1. The player encounters an auto pick-up item. 2. The player makes contact with item. 3. The player gains items. 4. Adjust player statistics according to item. 5. [Score] 6. [Health] 7. [Hit points] 8. The player continues with the level. |
| Post-Conditions: | 1. The current game continues. 2. The player statistics change. 3. The player’s inventory changes. |
| Error-Conditions: | 1. Controls are not responsive. 2. Controls lag. 3. Item does not load. 4. Item does not register as obtained. |
| Non-Functional Requirements: | 1. Controls must be responsive within .1 seconds. 2. Item must remain on-screen until next screen frame. 3. Item must have an equal chance of spawning. 4. Stats from item gain must update within .1 seconds |

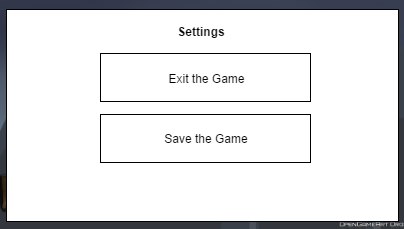
* 1. **Game mockups**
     1. Using the Main Menu



* + 1. Starting a Game



* + 1. Viewing Leaderboards/Progress
    2. Exit the Game (**In-Game**)

1. Settings Menu (**In-Game)**

* + 1. Encountering and fighting an enemy.

1. Enemy detects the player.



2. The enemy attacks the player.



3. The player selects a weapon and attacks the enemy.

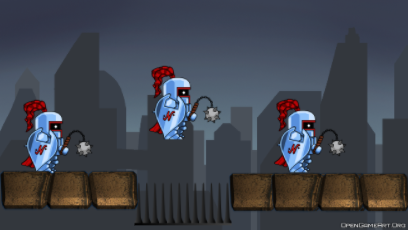


* + 1. Encountering obstacles and Platforms.

1. Encountering a gap and missing.



1. Encountering a gap and making it.



1. Encountering a platform.



* + 1. Encountering an auto pick-up item

1. Player encounters an item.



2. Player makes contact with item



3. Players stats change according to item type.



* + 1. Encountering a pick-up item

1.Player encounters pick-up item weapon



2. Player click key to pick up the them.



3. Item will change its respective stat. 