Zaiden Espe

1. **Introduction**:

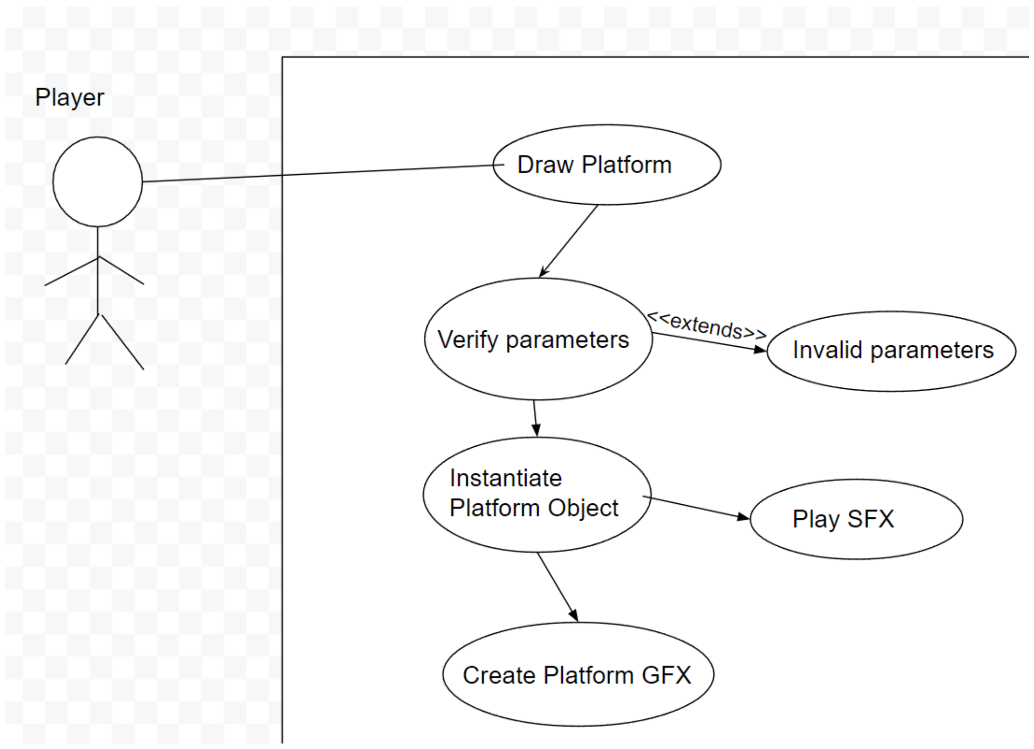
My feature is the platform drawing mechanic. The user's primary method of interacting with the game will be to strategically draw platforms for the hero character to traverse. The hero character will run forward on its own, making periodical jumps, and it will be the job of the user to draw platforms that let the hero avoid pits, jump over walls, and block/kill enemy NPCs.

I will need to make sure that the platforms are placed correctly on the map where the player draws their rectangle, and are of the correct size. I will also need to ensure that the platforms do not exceed specific size and area limits, and that the player has the required resources to create the platform they draw. I will also need to make sure the platforms trigger events in the NPCs, although it will not be my job to program how the NPCs respond to the platforms. I will need to make sure that the platforms drawn also correctly affect the platform drawing resource (which will probably be sand, dirt, or gravel). I will need to make sure that the platforms are correctly animated, meaning that the tileable sprites are combined perfectly to visualize the platforms. I also will need to find and implement the sound effects for when the player draws a platform, fails to draw a viable platform, and when an NPC collides with a platform.

Finally, I will need to implement two types of platforms: falling and floating. I will also need to ensure that the correct type of platform is made depending on what the player draws. I will need to create a system for how the player draws the platforms and chooses which type of platform they draw. The falling platforms will fall down until they go below the map, or connect with solid ground. The floating platforms will stay in the place where they are drawn.

2. **Use Case Diagram With Scenario**

Use Case Diagram:



Scenario:

Name: Draw platform

Summary: The player draws a rectangle on the screen to draw a platform. Then the system checks whether the player drew a viable rectangle by checking resource amount, rectangle volume, rectangle dimensions, and rectangle position. If the drawing is successful, the game system creates a new rectangle object with the correct parameters. Finally, the system creates the GFX and SFX for the rectangle.

Actors: Player

Preconditions: Level has been started and game is running

Basic Sequence:

Step 1: Player clicks and drags to draw a platform

Step 2: Level system stores the location of the initial click as the top left corner of the platform and the location of where the player releases the left mouse button as the bottom right corner of the platform.

Step 3: Level system calculates the area, and makes sure that the player has enough resources to build the platform, that the platform does not go over any pre-built level-features, and that the platform dimensions and area are within the acceptable limits.

Step 4: The level system instantiates a platform object with the correct input dimensions and position

Step 5: The platform object creates its visual component at its location on the map and plays a sound effect.

Exceptions:

Step 1: The level system does not instantiate a new platform object

Step 2: The level system players an error sound effect and displays an error message in the attempted platform location stating the determined error that disappears after 1 second.

Post conditions: A platform object now exists on the map, and depending on the input will either fall or float in place until the player moves significantly past the object or the level is completed.

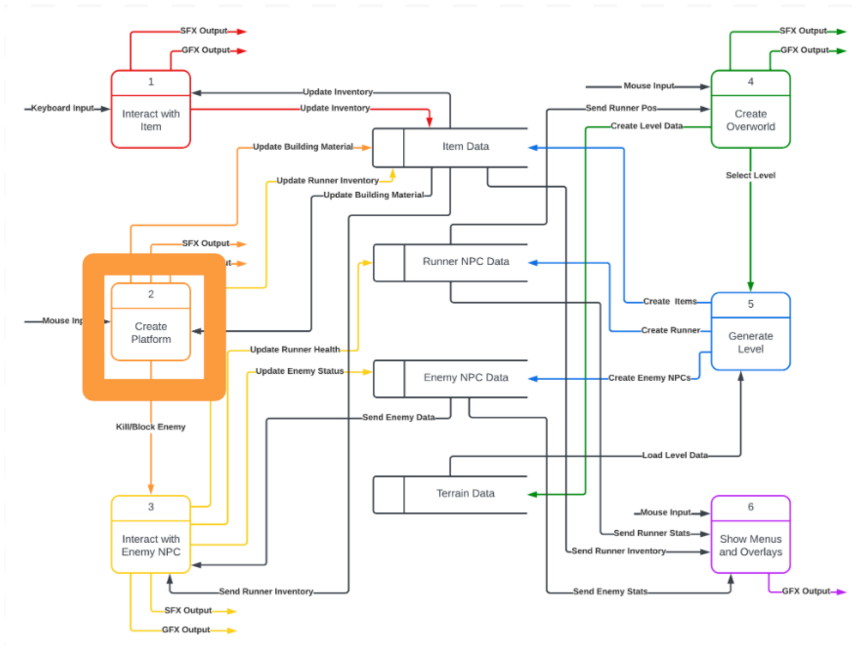
Priority: 1*

The priorities are 1 = must have, 2 = essential, 3 = nice to have

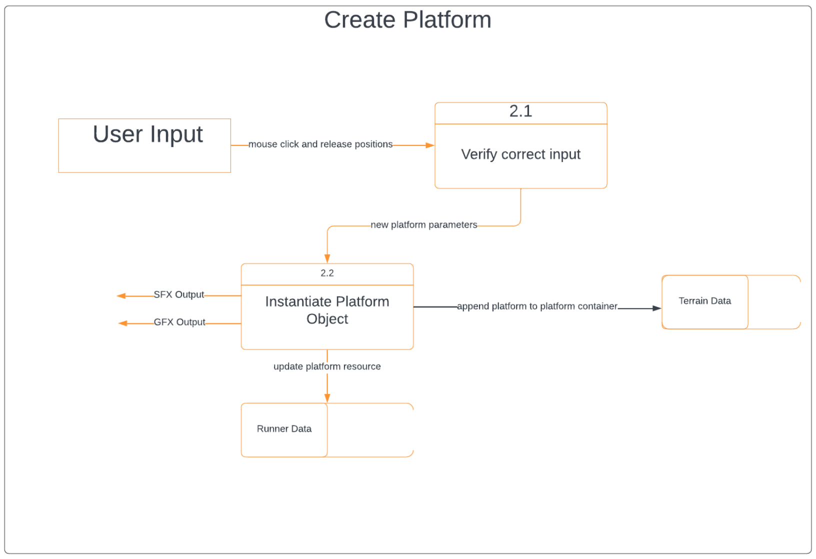
3. Data flow diagram(s) from level 0 to process description for your feature

Data flow diagrams

Level 0:



Level 1:



Process Descriptions:

2.1 - Verify Correct input

```
IF 1 < horizontal length < 5 THEN
IF 1 < vertical length < 5 THEN
IF horizontal length * vertical length < 15 THEN
IF horizontal length * vertical length < total platform resource THEN
IF position not overlapping other platforms or terrain THEN
Verify valid input
Move to instantiate platform step
ELSE
Play error sound effect
Display error message
END IF x5
```

4. Acceptance Tests

Input Validity Accuracy Test:

1000 random inputs will be given to the input validity function. The inputs will consist of an array of three tuples containing two coordinates and an integer between 0 and 100. Each tuple will have a value for the x-axis position and y-axis position. The first tuple will represent the point when the mouse was clicked, and the second tuple will represent the point when the mouse was released. The three tuples each represent a platform. The final integer represents the amount of platform resource the player has.

The output will be an integer that represents how many of the tuples representing platforms have valid inputs. Valid inputs will be determined by whether each platform has an area of no more than 15 square game units and no more than the amount of platform resources available to the player. The dimensions in the y-axis and x-axis must be greater or equal to one game unit but less than or equal to five game units. The inputs will also need to be drawn in the bounds of the gamescreen. Also, the corresponding rectangles to each input tuple will not be able to overlap each other. If there is overlap, neither platform will be valid. So, the possible outputs are 0, 1, 2, and 3. The output will also contain a boolean for each input platform that determines whether it is valid or not. True for valid, false for invalid.

The function will need to be successful for every input to be considered successful.

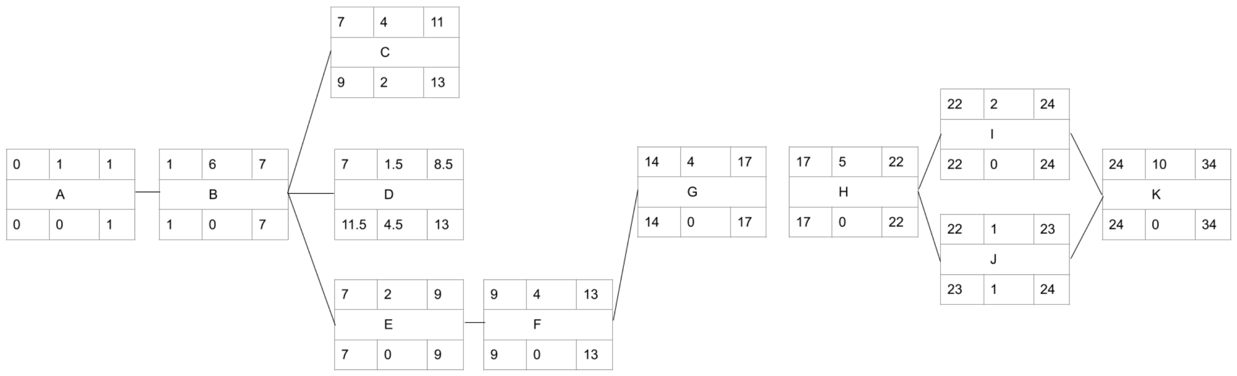
5. Timeline

Work items:

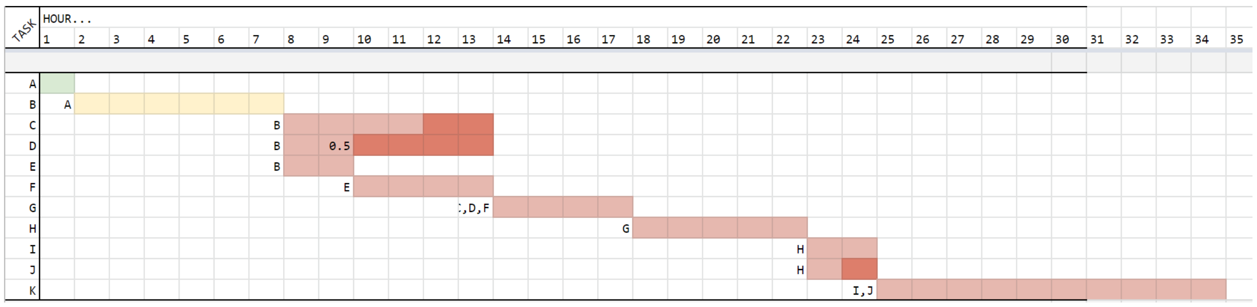
Task	Duration (hours)	Predecessor task(s)
A	1	N/A
B	6	A
C	4	B
D	1.5	B
E	2	B
F	4	E
G	4	C,D,F
H	5	G
I	2	H
J	1	H

K	10	J,I

Pert diagram:



Gannt Timeline:


TASK
HOUR...
A
B
C
D
E
F
G
H
I
J
K
A
B
B
B
E
:,D,F
G
H
H
I,J