Project LeapFall

Leap Motion Controller

Learn more...

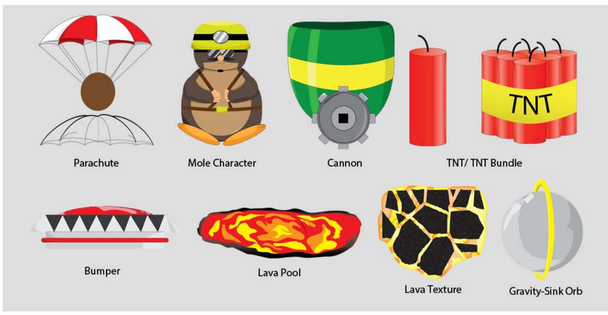
Our team utilized the Leap Motion controller and the Leap SDK within the game. Gameplay and the characters are controlled using input from the Leap device and translating it to Unity position data. The educational benefit of using an experimental device is that we had to rethink game controls in terms of hand/finger movements as opposed to classic keyboard/mouse movement. We built our game around the novel hand movements we thought up during the initial stages of the project.

2D Architecture

The 2D gameplay is accomplished using an orthographic camera showing the XY plane. Depth (z direction) is used to overlap layers on top of each other. All player movement occurs in the XY plane.

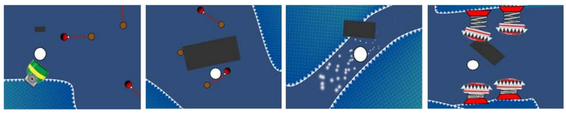
Art

I created most of the art assets for the alpha version of the game. The vector art was created in Adobe Illustrator and implemented as textures within Unity.



Interactive Obstacles

Most of the interactive obstacles are correlated with a texture above. I leveraged the NVidea PhysX physics engine built into Unity to allow real-time collision and create a dynamic environment. The main character as well as most of the obstacles are equipped with rigidbody add-ons that allow the object to behave according to the laws of physics. Values like force, velocity, and acceleration come into play and I manipulate these values to serve our gameplay goals.



The cannon obstacle allowed the charcter to land in the barrel, choose a direction in which to launch the character, and then shoot the character out at a high velocity.

The "ball on a chain" obstacle swings around and carries a solid "ball" at the end of a chain which can collide with a character flying by. A constant torque is added to the obstacle which allows it to spin in a circle.

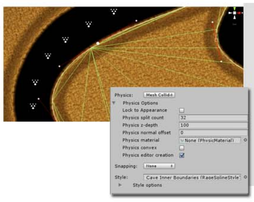
The "wind zone" acts like a wind tunnel which pulls the character along certain vectors at a given speed. A constant velocity force is applied to the rigidbody componenet of the character.

The "spring bumpers" are passive obstacles that deflect characters as they collide with them. I constructed a spring force system using hinges and spring components built into Unity.

Other obstacles included:

- Gravity sink: if the character flies too close to an orb, he/she will be "sucked" in towards it. It can be used to turn tight corners or it can slow the character down by pulling it up.
- Molten lava spouts: these spouts shoot out molten lava balls that will explode after 3 seconds of being created. The explosion emits a force vector in a 360 degree direction and will effect the character if it is too close.
- TNT sticks: single or bundles of TNT can be found in the cave and will explode on contact with another rigidbody object. The explosion will emit a force that will effect nearby players.

Collision Detection



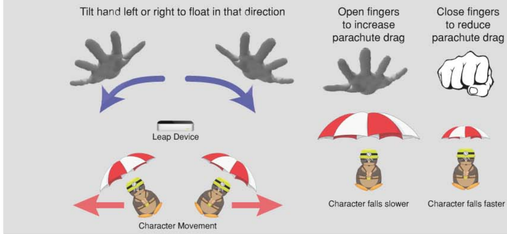
One of my primary roles included implementing and debugging collision detection between the player and environment. I utilized RageSpline's mesh collider functionality to create a boundary around the walls and environment obstacles in the level. The rigidbody component on the character recognizes these boundaries and collides properly.

The most frequent bug occured when the character would be propelled with a high velocity (by an obstacle or otherwise) and the collision detection between character and wall would fail, resulting in the character flying out of bounds and into empty space.

To fix the bug, I had to be sure the collision detection was "dynamic" and "continuous" to allow for more collision checks per update cycle.

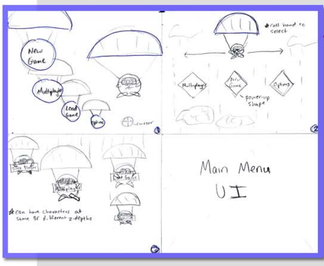
Character Control

Primary Movement Controls:



User Interface

I was tasked with mocking up the user interface for the main menu of the game. I started with hand-drawn wireframes and then used Illustrator to create "higher" fidelity designs. From there, the development team and I decided on a version to move foward with.



Above is the design we decided on.

The user tilts their hand left or right to select an option. The character will move in addition to a visual indicator of the current selection.

Level Design

I was in charge of prototyping the cave level layout. I broke the level into modules with different interactive obstacles in each. Below the modules are side by side but in the game, they would be stacked on top of each other. I implemented sections of the level by making "pre-fabs" and dragging them together like a puzzle piece.