CS4483 Prototype README

Built using Unity. An executable version is included in the project folder.

Controls:

A/D -> Movement left and right.

Space -> Jump, hold for higher jump.

Left Mouse -> Fire grapple hook.

1, 2, 3 numeric keys -> Select/Unselect an existing grapple.

Right Mouse -> When only one grapple is selected, holding allows you to connect it to a different surface.

Escape -> Open/Close start menu.

Q/E -> When one or more grapples are selected, increase and decrease length respectively.

R -> Delete any highlighted grapples.

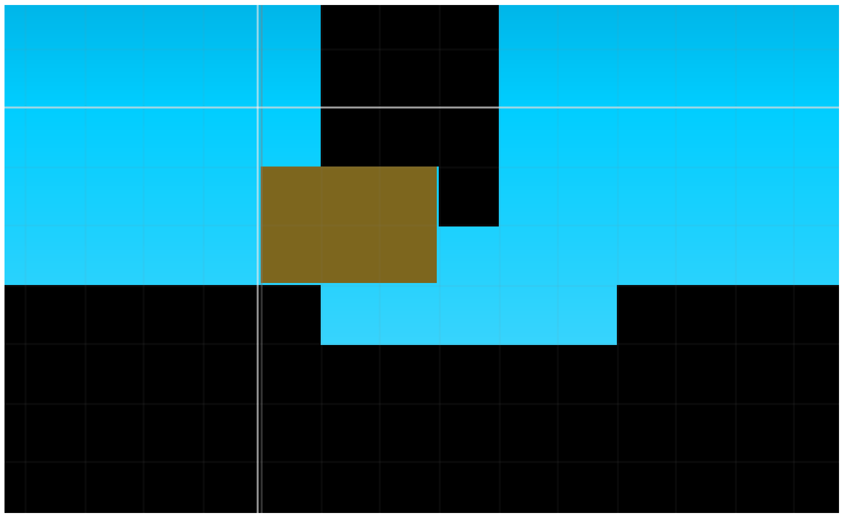
L -> Holding L for long enough resets the level and all objects.

Features:

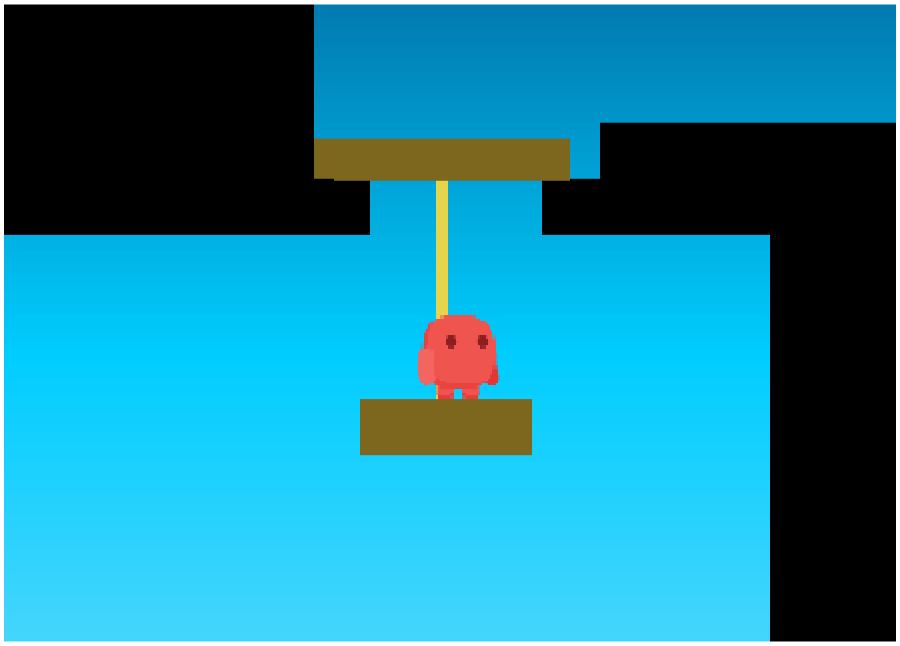
- Animated player character.
- Main Menu with Play and Quit Buttons
- Start Menu in-game to return to the main menu or resume.
- Grapple mechanic with rope-like feel.
- Toggle between active grapples and increasing/decreasing their length.
- Player can use these grapples to swing from the map and pull around objects found lying on the ground, or suspend them from the map.
- A few basic puzzles exist, described below.
- Level reset feature via holding the L key.
- Red blocks can’t be grappled, but are part of some core level objectives.

Puzzles

This one is simple, just shoot a grapple at it and pull it out of the path.

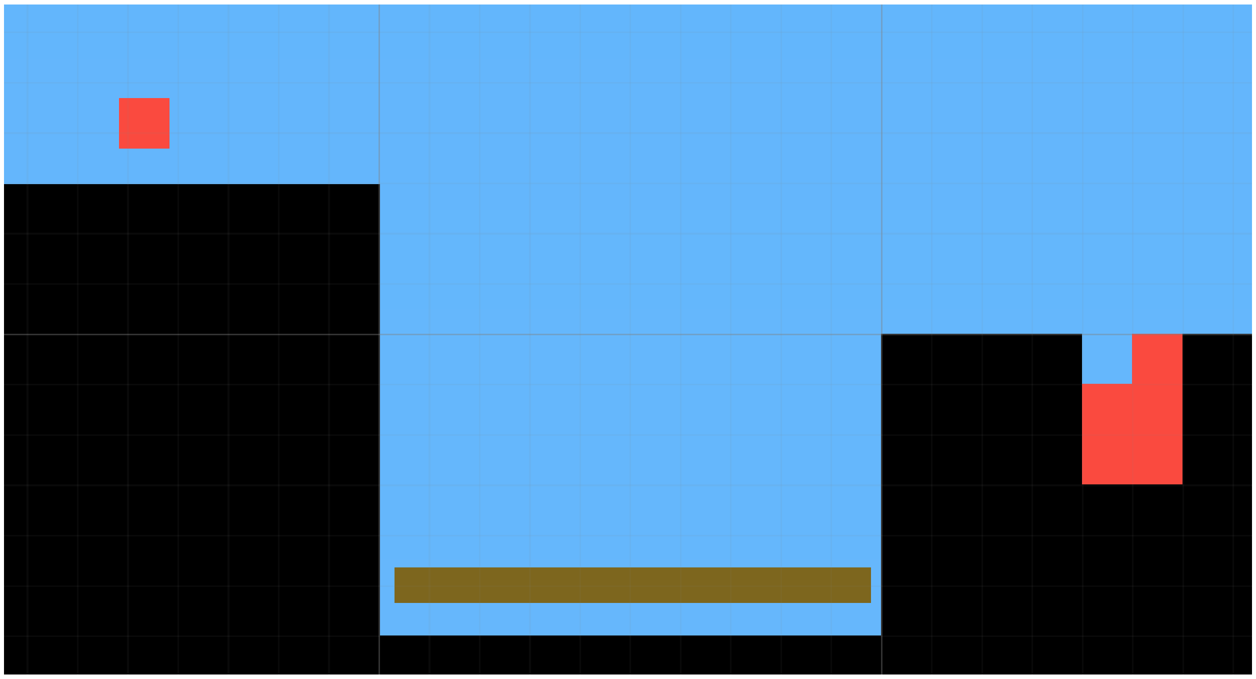


For this one, you have to attach the block on the ground up to the plank blocking the way up. Decrease the grapple length with E (after highlighting) to get closer from on top of the box. At this point, you need to quickly delete the rope (R key) and jump up into the plank to push it out of the way and get up.



Red objects represent objectives and can't be grappled. The goal here is to get the box into the hole where it belongs. It can't fall in the pit between or it will respawn. Shooting a grapple down at the plank and connecting it to the roof (offscreen in that picture) is how you go about raising it (with E after connecting it to the roof). Then push the box across. **The grapple and object masses aren't perfectly tuned, and so some weird behaviour can happen when "reeling in" a rope, and it comes off as being delayed a bit. This can be mitigated by just extending the rope bit by bit until it's the right length.**

The plank and red box normally fall to the ground when the scene runs.



The final sequence is just to swing around with the grapple.

I purposely didn't have the level quit back to the main menu on reaching the end so the player can continue messing around in the prototype.

The main menu Level Select button was left non-functional on purpose, but would be present in the full game.

CS4483 Game Project

Save Data

My thoughts on implementing save data is to use a "SaveDataManager" gameobject that persists between scenes. All it needs is a singleton script attached to it of the same name that is able to load and save data to a file contained within the project. The data itself will more or less just be an array of float values, which represent the best timed score of each level at their respective indices. Load can occur once upon the game launching, and saves occur when a new level record is set.

A couple useful links:

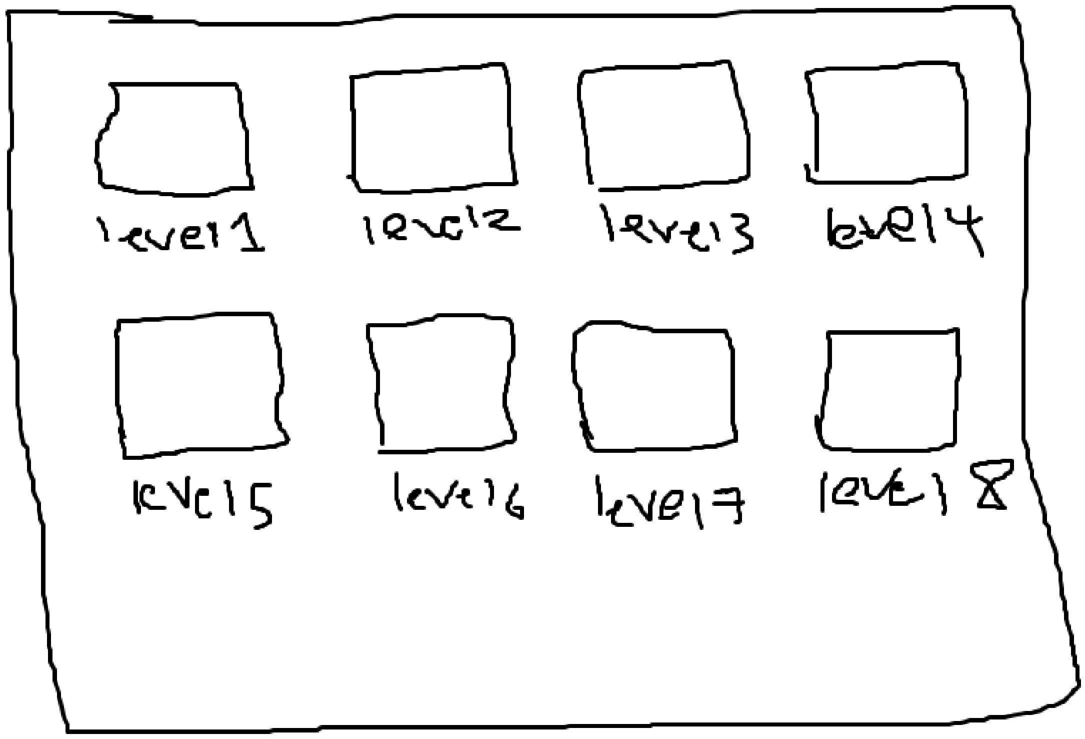
Save Data (scroll down to "Saving The Game" section): https://www.raywenderlich.com/418-how-to-save-and-load-a-game-in-unity

Singleton GameObject: https://gamedev.stackexchange.com/questions/116009/in-unity-how-do-i-correctly-implement-the-singleton-pattern

Level Select

The button to access level select is currently there on the main menu but isn't hooked up to anything. I imagine it working a bit like this: just like actual gameobject prefabs, it's also possible to create prefabs for UI groupings. For example the level select screen could be a table/grid looking view where each cell is a prefab containing an image of one part of that level, with a label below it saying *level x*. I use a panel for the container of the main menu UI, and similarly a panel could be used as a container for the level select "grid". There's a UI component that can be attached to the panel directly called “Grid Layout Group”, which nicely formats any children of the panel according to the settings in that component.

The end result might potentially look like this (Incredible MS Paint skills):



Where each image in the prefab can have a button component added to it, and also a small script that defines a function to load the proper scene. This function also has to notify the save data manager which index it is, so the manager knows where to save the score to upon completion.

Level Timer

The level timer can just be an empty gameobject in the scene with a timer script on it. Upon level completion, the timer script sends its time over to the Save Data Manager singleton instance, where the score is deemed worth saving or not on the saving-end of things. I’m 99% sure pausing will already stop the timer because it reduces the timescale to 0 so don’t worry about that.

Level Creation

Making levels using the tilemap system is pretty easy. First, you have to create a new tilemap object in the new scene you’re working in. Then, you have to open another window call “Tile Palette”, and select which palette you are currently working on. You can create Tiles out of existing sprites and drag them onto any one of the palette grid cells in the palette window. Then, you can select any of the drawing methods and begins painting the tiles directly onto the scene. One of the last things you have to do is add a Tilemap Collider to the tilemap gameobject, so the player won’t fall right through. Lastly, change the layer, to ground or map layer. Whatever the option is I added there.