**Team Name: Lemonlight Studios**

**Alex Karpyuk (Team Leader)**

My role in this Milestone was the logic in game itself. I made system with States, that changes current state of the game because of the situation(condition) in the current state or input of the player. For example, for now, the winning condition in the game is to collect all carrots on the level. However, in the future we will change it. In addition player will have to take its character to specific place at the screen. For now, I did not find any kind of bugs with states.

Also, I implemented carrot system. Carrots are one of the winning conditions for level. So if player intersects with them, they disappear from screen and level score updates.

Next, I made spikes. At this stage, spikes are objects that kills player instantly, if he intersects with them. They spawn at the beginning of the Building phase, so player can see where they are.

With spikes, I implemented health system and screen bounds. If player intersects with spikes, or falls out from the bottom edge of the screen - game is over. If player intersects with left or right bound of the screen, then they can not move out of screen.

I did not find bugs within stuff I done yet. We will ask our friends to play game and try to find bugs. Overall, the project did not have changes for now. Everything goes on time.

**Max Bennett (Design)**

My role in this milestone was making the movement of the player and allowing the platforms to be dragged across the screen.  I created the players gravity, hspeed, vspeed, and acceleration attributes, and handled the player jumping when the up key is pressed.  The acceleration and gravity are set up to make the players movement feel smooth. I worked with the platform class to set up a system where only one platform can be dragged at a time and gave platforms an attribute that determines if it is draggable or not.  This will be used to differentiate platforms that the player can move and platforms that are set by the level.

There were bugs with the movement when integrated with the collision code, such as the player falling through/moving through platforms occasionally.  No observed bugs with moving platforms yet. We need to find a good way to control when a player can jump and when they cannot.

**Max Hazel (Architect)**

My job was to make the level class, file reader, and external tool for creating and loading separate levels.  I also improved collision detection to work on any object in the level. I made a Level class, which is an object that can hold the amount and positions of objects in a single level, and can be loaded using information in external text files.  I added a method to Game.cs that can read and store data for a level from properly formatted text files. The external tool takes user input and uses it to create one of these level files, though currently there is no way to check that the user’s input is properly formatted, and the text file must be moved manually.

Bugs:

Player can jump infinitely, and can clip through platforms while jumping.

Carrots do not load when restarting a level, need to make them visible again when a level is reloaded.

External tool does not check to see if user input is correctly formatted; text instructions are provided in the window.

**Kimmy Goewey (UI)**

My role was to implement a working UI for the game.

I created a game over, main menu, pause, and nextLevel menus along with background images and carrots. Currently, I want to add the items list that I had mentioned in milestone one.

My job was to make sure everything looked good and ran fluidly. I also fixed our repository.