**Change History Documentation**

|  |  |  |
| --- | --- | --- |
| **Date** | **Changes Made** | **Reasoning** |
| January 27th, 2017 | Created the basic framework for the game engine that will be running the applications behind the scenes. | The game engine needs to be created first in order to create objects that can be developed for the game. |
|  | Created a basic renderer that is able to render objects and images to the screen. | Rendering objects to the screen is required. |
| February 17th, 2017 | Updated the coordinate system to be relative to the screen instead of absolute positioned | Relative positioning allows the ability to add rooms/maps that have coordinate systems which are larger than the screen. It also allows future development for camera perspectives and mini-maps. |
| April 27th, 2017 | Implemented various types of cameras that the user is able to customize. | Cameras allow the user to create a top-down vs. side-scroller, and other types of games by allowing custom camera perspectives on the game. |
|  | Created a README for a public-facing description of the project | Allows public visitors to see the goals, and objectives for the project. |
|  | Added additional camera configurations | With additional camera configurations, the user is able to select from a few pre-set camera options without having to configure all of the details themselves. However, the option to configure the camera will still be available. |
| April 29th | Added the ability to create different screens, a screen manager, and guis. | Giving the user the ability to change the view of the screen is required in game development as cut-scenes, camera pans, and GUIs are essential to many games. |
|  | Updated the SpriteLoader module to only load sprites when they are required by the game. | This prevents the game from having to load all of the required resources at startup and result in long loading times. |
| May 4th | Started working on the user interface which will allow the user to create their own game. | The user cannot create their own game with only a game engine. The user interface is the user’s primary method of development and is a requirement for the user to create their own game. |
|  | Added Modals which can prompt the user for information when user input is required before any action can be taken. | Modals allow the user to input the name of their objects that are being created and can alert the user of dangerous operations such as deletes and prevent the user from accidentally removing all of their work that has been created. |
| May 10th | Started work on the development Engine. A ResourceManager module keeps track of all of the resources that the user has created. | The development engine is the engine that will run behind the user interface. When the user creates an object, it is not immediately added to an instance of the game because that would cause the game to be constantly running while the user is developing the game. The Development engine keeps track of all of the user’s requirements and settings for the game until the user is ready to deploy their game. |
|  | Added Right-click menus | Added ease-of-access for super-users who would like to create resources faster. Instead of mousing to the menu bar each time, the user can right click and create a new resource immediately. |
| May 14th | Started development on the sprite, object, script, and room editors | These are the primary editors which will allow the core functionality of the game. These modules are user-interfaces which interact with the Game Development module in order to allow the user to customize the components of their game as they would like. |
| May 15th | Added a documentation section to the github repository in order to track planning, ideas, and thoughts while developing the project. | This folder allows me to keep all my documents for planning and developing the project in the same location. Also gives me a better idea of how to create the project by following an outline of planned features. |
| May 20th | Added Properties which save the custom properties about each object that a user will edit. Properties also implement the observer pattern which allows user interface items to update their display whenever a setting is changed by the user. | The addition of properties to the game provide better control over the customizability of options that the user has. Through the user of the observer pattern, UI elements can each be a PropertyObserver in order to observe the property that they are concerned with customizing. This allows the user to immediately see any changes that they make in the user interfaces. |
| May 25th | Fixed some bugs by adding Observer propagation | Some resources that the user can create will have the ability to link to other resources. For example, an object will be linked to a sprite in order to be linked to the image properties. However, if the sprite was updated this would not update the object. Adding observer propagation, I allow linked resources to send out notifications when any of their linked resources are changed as this may affect the user interface. (If a sprite’s image is changed we don’t want the image to still display the old one in the object settings!) |
| May 26th | Added drag-and-drop functionality to the room editor which allows the user to drag objects from the list of objects into the room to place them. | Drag-and-drop functionality allows the user to better choose where the objects they are placing should be placed within the room. By using a canvas to draw the items, this also allows for advanced functionalities like checking for right or left clicks on an object’s bounding box once it is inside of the room editor to provide the user with advanced options such as deleting and fine-tuning the positioning of the object in the room. |
| May 27th | Added sprite origin and image size settings into consideration when placing an object inside of the room. | The user is able to scale the size of their sprite as well as select its origin in the sprite editor. These properties are taken into consideration when placing an object into the room and will properly place the object’s origin where the mouse is released. |
| June 3rd | Added the ability to scroll, zoom and control the camera in the room editor | This allows users to zoom in or out of the room in order to refine small details while also being able to manage objects at a macro scale. |
|  | Added the ability to delete objects once placed in the room | If the user makes a mistake, they don’t want the mistake to be un-doable. By giving the user the ability to delete objects in the room, they can correct any mistakes they may have made. |
| June 18th | Allowed user creations to be placed in the game when building | By placing the user code into the game, We now have a functioning game engine as the code that the user develops is actually visible and testable as an application. This allows users to be able to test the objects they have created and make necessary changes to the game |
|  | Mediocre refactor of the game engine | In order to be able to use code form the Development Engine in the game engine, A minor refactor of the game engine was required. The refactor now allows the game engine to create the objects needed to run the game from the already previously created development engine resource objects. |
| June 29th | Start to work on dynamic code compilation | In order to parse the user’s desired actions for moving the players, and interacting with the game, dynamic code compilation is required. Began to create a compiler which will compiled user-entered code into insatiable java classes. |
|  | Added a library which allows syntax-highlighting in text-areas for code editing. | Having syntax highlighting and code checking for the users is extremely useful as when they write their own code to make the objects interact, they will have to be writing in java (for now) . Having a text box which can format code helps the programmer and is essential for a game developer. |