Sanctum of the Chalice

Sprint 1 planning document

Team 28:

Phoebus Yang
Shubham Jain
Taehoon Kim
Alec Hartline
John Pesce

Scrum Master: Phoebus Yang

Meeting Times: Mondays & Friday 5PM

Sprint Overview:

During this sprint, we hope to create a minimal viable product that showcases our product's potential. The goal of this sprint is to design our system such that it will make adding future content easier. We plan to create base versions of the rendering engine, game engine, level generator, sound engine, and object engine to produce a playable tutorial. The tutorial will contain and basic procedural level generation as well as instructions for the base controls of the game such as player movement.

Risks and Challenges:

The greatest challenge will be setting up the base of the entire system as multiple engines rely on each other to function correctly. Furthermore, laying the foundation of each engine is a task of sizeable proportions. Tackling this problem will consume a lot of time without a lot of visual output. Communication will be key in implementing these tightly intertwined engines. Another challenge will be ensuring our data structures are safely synchronized without drastically slowing down the game, since we do plan on running three different threads. We will have to pay close attention to avoid race conditions and other synchronization problems.

Sprint Overview:

Story #1:

As a developer, I would like for a clear and hierarchical class system that can represent attributes and relationships of all game objects in a logical manner to exist

Task	Time estimate	Owner
Develop class system to represent gameobjects and their attributes	4 (2 each)	Alec, John
Implement abstract class functionality	5	John
Add class to handle all Game Engine requests	4 (2 each)	Alec, John

- Given that the class system is sufficiently robust, when adding future object classes, there will be no need to refactor existing code

- Given that the abstract class system is implemented properly, when working with the JSON handler, the abstract classes will encompass the structure required by the JSON handler
- Given that the request handler functions correctly, when the game engine requests a gameobject, the object engine will quickly initialize and return a copy of the appropriate template

Story #2:
As a player, I would like to have a tutorial that explains and utilizes core game mechanics

Task	Time estimate	Owner
Bring together all system components of the game to produce a tutorial level	7	Shubham
Choose non-complicated words for introducing game mechanics	1	John
Implement instruction messages that appear in the game window	2	Taehoon
Pick out sprites and assets to use for first iteration of tutorial	3	Taehoon
Pick out music for first iteration of game	2	Taehoon

- Given that the central runner for the tutorial is implemented correctly, when the player opens the game, they get a fresh procedurally generated level, and can move around the level and learn about the base game mechanics
- Given that the text for all initial interactions is added in, when the player encounters any help text, it is simple enough for them to get a basic grasp of the mechanic being explained
- Given that the instruction messages are properly implemented, when the user encounters content that requires new game mechanics such as movement or fighting with enemies, appropriate help instructions will popup
- Given that sufficient sprites and music have been chosen for the tutorial, when the tutorial is created, there will be enough sprites to represent gameobjects and music to play in the background

Story #3:
As a developer, I would like to have a rendering engine that can process and render sprites or sprite-sheets as PNGs to the screen

Task	Time estimate	Owner
Write class to load png images as sprites	5	Alec
Create algorithm to process spritesheets and handle assignment of assets	5	Alec
Create the game window and a mechanism to draw sprites onto it	4	Shubham
Develop class to interact with Game Engine	5	Shubham
Make interaction class thread-safe	5	Alec
Add support for rendering engine to draw text on screen	3	Shubham

- Given that the draw function is written correctly, when the rendering engine calls the draw function, the game window will be updated to reflect the current game state
- Given that the rendering engine is implemented well, when running the game, the rendering engine will be able to paint at least 30 frames per second
- Given that the algorithm for processing sprite sheets works as intended, when the sprite loader reads in a sprite sheet, the algorithm will crop the sheet into 32x32 pixel area units
- Given that the interaction class is made to be thread-safe, when the game engine and rendering engine communicate, they will be able to act independently of each other while avoiding race conditions
- Given that the text rendering algorithm functions properly, when the rendering engine receives a text request, the text will be printed to the screen at the designated location

Story #4:
As a developer, I would like to be able to read in sound files in .ogg format and associate them to either music or sound cues

Task	Time estimate	Owner
Learn how to use the OpenAL sound interface	4 (2 each)	Phoebus, Taehoon
Develop a system that can read in .ogg files	2	Phoebus
Implement functions that can play a sound file given a variety of parameters	4	Phoebus
Implement a sound loop for background music along with a function to terminate the loop	6	Taehoon

- Given that the play function is properly written, when the function is called with a file name, length and volume parameters, the corresponding .ogg file will be played for the appropriate length at the appropriate volume
- Given that the sound loop is implemented correctly, after the sound loop function is called, the passed in file will play in a smooth loop indefinitely
- Given that the sound loop termination function works as intended, when the termination method is called, any looping music will stop playing

Story #5:
As a developer, I would like to have a game engine that facilitates updating the states of game objects

Task	Time estimate	Owner
Write a loop that facilitates rendering, updating, handling user input, and any other essential functionality	7	Phoebus
Implement a slow tick system to represent the pacing of the game	4	Phoebus

Create algorithm to determine objects in player vicinity	4	John
Implement user input handler	4	John
Write function to update state of valid objects	6 (3 each)	Phoebus, John

- Given that the tick systems are properly implemented, while the game is running, the fast system will be called 30 times a second while the slow system will be called twice a second at most
- Given that the function for getting relevant gameobjects works properly, while the game is running, it will contain a list of all gameobjects within a certain radius of the player.
- Given that that the relevant gameobject action function is built correctly, each time the slow tick occurs, the action functions for all relevant gameobjects will be called and resolved.
- Given that the fast update function works properly, each time the fast tick is called, gameobjects will be updated to intermediate states between starting and ending the action specified by the previous slow tick.

Story #6:
As a developer, I would like to have a game loop that calls the appropriate engines (rendering, level generation, etc.) when required

Task	Time estimate	Owner
Implement class to handle rendering engine interactions	4	Phoebus
Implement system to retrieve information from Object class system	2	John
Implement logic for determining level to be generated	2	Phoebus
Implement system to send requests to sound engine to play music and sound cues	3	John

- Given that communication between the game and rendering engines is set up properly, for each fast tick, the rendering engine will be sent a request to update the game view in a thread-safe manner.
- Given that communication between game and sound engines works correctly, when events with associated sounds occur (such as starting a level), an appropriate call to the sound engine will be made.
- Given that communication between the level generator and game engine is implemented well, when the level begins, the game engine will request and receive a map from the level generator.
- Given that communication between the game and object engines is established correctly, when a new gameobject needs to be created, the game engine will get the appropriate template from the object engine.

Story #7:
As a developer, I would like to have a physics system that allows for player movement and collision detection

Task	Time estimate	Owner
Implement collision detection between solid game objects	6	Alec
Add functionality to allow player to move through certain blocks but still trigger relevant events	4	John

- Given that the collision detection system works as intended, when an entity attempts to move onto a solid block, it will be prevented from doing so
- Given the collision event system is implemented correctly, when an entity occupies the same tile as an entity with a special interaction, the proper event will get triggered
- Given that the collision detection system functions properly, when an entity looks ahead at possible movements, it will be able to see if the attempted movement would result in a collision

Story #8:
As a player, I would like to be able to navigate a main menu so that I may start a new game or change game settings

Task	Time estimate	Owner
Implement a basic welcome screen	2	Shubham
Implement sound control button to change the volume	2	Phoebus
Implement different scenes of rendering menu windows	3	Alec

- Given that the welcome screen in implemented correctly, when the game is opened, the welcome screen will appear
- Given that the sound control button functions properly, when the sound control is manipulated, the music and sound volumes are adjusted accordingly
- Given that the start algorithm works as intended, when the game is started, the level generation functions will be called

Story #9:
As a player, I would like to be able to experience procedurally generated levels

Task	Time Estimate	Owner
Develop algorithm to generate a dungeon based on a random seed	7	Shubham
Develop individual rooms and corridors formats to fit into the dungeon	5	Taehoon

- Given that the generation algorithm works properly, when the algorithm is called, a dungeon matching the given parameters will be created
- Given that the generation algorithm is random, if a different seed is used, then the level created will be sufficiently unique.
- Given that dungeon components are designed well, when the level is generated, the components will fit together seamlessly

Story #10:
As a developer, I would like to have a procedural level generation system that can accept multiple parameters and generate unique, challenging levels.

Task	Time estimate	Owner
Implement algorithm to populate dungeons with items and entities	3	Taehoon
Implement auxiliary class to retrieve and sort through tile data from object engine	4	Taehoon
Design map standard for ease of communication with Rendering engine and Game engine	2	Taehoon

- Given that the dungeon population algorithm works as expected, when the level is created, the level will be filled with appropriate entities
- Given that the tile sorting class functions well, when the level generator is building the level, it will be able to find and lay appropriate tiles quickly and efficiently
- Given that the map standard is developed correctly, when the level generator creates a map, the game engine will be able to receive and extract all necessary data regarding the level from the standard

Story #11:
As a player, I would like to be able to control my character around the game world.

Task	Time estimate	Owner
Implement keyboard listener with proper input handling	2	Shubham
Write an auxiliary class to notify game engine of user input	3	Alec

- Given that the Keyboard listener is implemented correctly, when the user presses a key, the rendering engine will detect the event and note what key was pressed

- Given that the notification class works properly, when the rendering engine receives keyboard input, the game engine will receive data regarding the key pressed
- Given that the auxiliary class is made well, when the rendering and game engines communicate, concurrency issues will not occur

Story #12:
As a player, I would like to be able to view my overall score and achievements upon the completion of levels.

Task	Time estimate	Owner
Implement a level timer and visual display upon completion	1	Shubham
A system to keep track of player statistics and score	2	Taehoon

- Given that the level timer works as intended, when the level ends, the clear time will be presented to the player
- Given that the player statistic tracker works correctly, when the player performs specific actions, the score will increase
- Given that the player statistic display functions properly, when the level ends, a breakdown of the score will be shown

Remaining Backlog

Functional Requirements

1. As a developer, I would like for a JSON attribute system to exist, so that I may organize and sort game data
2. As a player, I would like to see a heads-up-display that gives me all relevant information in a clear format
3. As a developer, I would like to be able to edit the attributes of existing items and add additional items with ease
4. As a developer, I would like to have a robust serialization format to facilitate saving and loading game states
5. As a player, I would like to be able to save my game state and load it at a later time
6. As a player, I would like to see the amount of time before the next tick executes in a clear manner
7. As a developer, I would like to have a system that can keep track of a player's movements and be able to revert the player to a set limit of previous movements
8. As a developer, I would like to have a sound engine that can play both music in the background as well as sounds based upon actions being performed in game
9. As a player, I would like to hear distinct and clear audio cues for certain actions performed by myself or by the environment around me
10. As a player, I would like to be able to encounter unique puzzles every level and be rewarded for solving them
11. As a player, I would like to be able to use the time revert system to aid in combating enemies and overcoming traps
12. As a player, I would like to be able to combat threats and be rewarded for succeeding in overcoming them
13. As a player, I would like to be able to invest experience gained into stats to scale in power as the game gets harder
14. As a player, I would like to choose from multiple character classes and specialize in certain attributes to improve my character through the game
15. As a player, I would like to be able to collect and equip items or equipment and use them for combat and other purposes
16. As a player, I would like to be able to learn and use unique abilities that diversify combat and allow for interesting interactions
17. As a developer, I would like to provide players a challenge fighting enemies through a robust AI system
18. As a developer, I would like to be able to assign ability rotations to enemies to increase combat difficulty
19. As a player, I would like to be able to encounter and fight bosses that present opportunities to progress in the game

20. As a player, I would like to be able to gather experience by completing tasks like killing enemies and solving puzzles
21. As a player, I would like to be able to tweak game settings like difficulty and sound volumes
22. As a player, I would like to be able to personalize keyboard controls in the game settings
23. As a modder, I would like to be able to modify or add items to the game with ease
24. As a modder, I would like to be able to specify sprites or sprite-sheets for custom game objects
25. As a player, I would like to be able to meet friendly non player characters, and have unique interactions with them
26. As a player, I would like to experience random events that provide unique experiences each time
27. As a player, I would like to keep track of the bits of lore I have uncovered at any point so that I may put together pieces and understand the backstory (if time allows)
28. As a player, I would like to be able to take in-game screenshots of high quality (if time allows)
29. As a developer, I would like for a particle system to exist that adds to the ambience of the game and can be used for special indicators (if time allows)

Non-functional Requirements

Rendering

Being able to render multiple sprites and images to represent the various game objects in the game is an essential part of *Sanctum of the Chalice*. Beyond this, rendering animations on the screen is another important part. The Swing and awt libraries in Java are going to be used to implement the rendering engine. Images for sprite-sheets can easily be read in, processed, and passed into awt's Canvas.paint method to be displayed on a JFrame. We aim to secure at least 30 frames per second of frame-rate on mid-range PCs and laptops. We plan to achieve this by using rendering techniques like caching, and using lightweight image formats for our art.

State Updates and Input Processing

For player satisfaction, players must see actions they are trying to perform smoothly reflected in their game view. This is going to be facilitated in two parts. Firstly, to lighten up the update queue and thus decrease the number of iterations on average per game tick, only gameobjects and entities influencing the player will be updated at any given tick. Several games such as Minecraft and Terraria use this technique, where only objects in "chunks" near the player are updated every tick. This approach is geared towards minimizing CPU load, and allow for more resources to be spent on the graphics and sound threads. Secondly, player input must feel snappy for a smooth gaming experience. Since the game world itself will operate on a slow tick system, player inputs that are provided in between ticks will be buffered, and a proper visual indication of this buffering will be provided to the player.

Scalability

Since our game engine and rendering engine will be fairly independent of each other, we will be able to make changes to either without incurring any major changes in the other. Many games lack a clear system for modifying and updating game content. We will implement a JSON attribute system which will allow us to define the characteristics of every gameobject in segregated JSON files (for example, one for items, another for tiles, and a third for friendly NPCs and so on) and read these files while loading the game. We can then generate the actual in-game objects based upon the number of objects present within these JSON files. Not only does this make it extremely easy for us as developers to add and modify content without hard-coding every unique gameobject, but it also allows for "modding" by players, so that they may enjoy custom content or change current content to fit their specific tastes.

Sound

Background music and sound cues are essential for setting a proper ambient environment and for allowing players to process changes in the environment when these changes cannot necessarily be seen clearly. Java's native Sound library, while providing a decent variety of sound encodings, offers only limited support for lightweight file formats. To reduce resource consumption, we plan on making use of the Ogg Vorbis (.ogg) sound file format, which is a very lightweight format that is extremely popular in game development. The .ogg file format runs less than 128 kbits per second for medium quality. It provides comparable quality to an mp3 at a lower bitrate and a smaller file size. To play these .ogg files, we will use the OpenAL library for Java.