Sanctum Of the Chalice

CS 307 Team 28

Project Backlog

Presented by Team 28:

Shubham Jain

Phoebus Yang

Taehoon Kim

Alec Hartline

John Pesce

Problem Statement

Roguelite and roguelike dungeon-crawler games generally involve only stat-check turn based mechanics. This results in players not having strategize their journey through a dungeon. Our project aims to present lovers of the roguelike genre the old classical stat-based dungeon crawling experience but with the twist of unique strategic mechanics, along with a fusion of turn based and real time gameplay.

Background Information

Target playerbase

Modern gaming has become heavily diversified by a plethora of different genres, but older genres like roguelike and dungeon-crawlers have never failed at attracting a significant playerbase. However, these genres have for the large part failed to necessitate strategic thought and tactical prowess from their players. *Sanctum of the Chalice* aims to entice fans of the mature roguelike genre looking for something different, while at the same time setting itself just far enough apart from typical examples of the genre to provide a unique gameplay experience and thus bring in players of the strategy and RTS genres.

Similar games

The closest example of an attempt at blending the roguelike dungeon-crawler experience with another genre is *Crypt of the Necrodancer*. *Crypt of the Necrodancer* attempts to stay close to the roguelike genre by having procedurally generated levels with randomized enemies and loot, but strays away from the core of the genre by removing the classic stat based interaction system. It tries to approach the strategy and twitch style of play by introducing a song beat based tick system which adjusts itself according to the tempo of the background music.

Limitations

In *Crypt of the Necrodancer*, as the tempo of a song ramps up, the tick system starts ramping up as well. At later stages of a level, this requires players to make reflexive decisions about where they have to move and how they have to engage in combat. While *Crypt of the Necrodancer* tries to maintain a level of strategic gameplay requirement, the devolvement into a “twitch” style game leads to a complete dissolution of any tactical requirement from the player. *Sanctum of the Chalice* will instead operate on a fixed tick system that will leave the player enough time to consider possible strategies while maintaining some degree of urgency. Furthermore, *Crypt of the Necrodancer* lacks stat based and scaling mechanics which are essential to a game being labeled as a roguelike in genre. In contrast, *Sanctum of the Chalice* will contain classic elements of the roguelike experience such as stat checks, equipment, and a loose class system.

Functional Requirements

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
2. 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
3. As a developer, I would like for a JSON attribute system to exist, so that I may organize and sort game data
4. As a player, I would like to see a heads-up-display that gives me all relevant information in a clear format
5. As a player, I would like to have a tutorial that explains and utilizes core game mechanics
6. As a developer, I would like to be able to edit the attributes of existing items and add additional items with ease
7. As a developer, I would like to have a robust serialization format to facilitate saving and loading game states
8. As a player, I would like to be able to save my game state and load it at a later time
9. As a developer, I would like to have a physics system that allows for player movement and collision detection, as well as projectile motion
10. As a player, I would like to see the amount of time before the next tick executes in a clear manner
11. 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
12. 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

13. 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
14. 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
15. As a developer, I would like to have a procedural level generation system that can accept multiple parameters and generate unique, challenging levels
16. As a player, I would like to be able to experience procedurally generated levels
17. As a player, I would like to be able to encounter unique puzzles every level and be rewarded for solving them
18. As a player, I would like to be able to use the time revert system to aid in combating enemies and overcoming traps
19. As a player, I would like to be able to combat threats and be rewarded for succeeding in overcoming them
20. As a player, I would like to be able to invest experience gained into stats to scale in power as the game gets harder
21. As a player, I would like to choose from multiple character classes and specialize in certain attributes to improve my character through the game
22. As a player, I would like to be able to collect and equip items or equipment and use them for combat and other purposes
23. As a player, I would like to be able to learn and use unique abilities that diversify combat and allow for interesting interactions
24. As a developer, I would like to provide players a challenge fighting enemies through a robust AI system
25. As a developer, I would like to be able to assign ability rotations to enemies to increase combat difficulty
26. As a player, I would like to be able to encounter and fight bosses that present opportunities to progress in the game
27. As a player, I would like to be able to gather experience by completing tasks like killing enemies and solving puzzles
28. As a player, I would like to be able to view my overall score and achievements upon the completion of levels
29. 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
30. As a player, I would like to be able to tweak game settings like difficulty and sound volumes
31. As a player, I would like to be able to personalize keyboard controls in the game settings

32. As a modder, I would like to be able to modify or add items to the game with ease
33. As a modder, I would like to be able to specify sprites or sprite-sheets for custom game objects
34. As a player, I would like to be able to meet friendly non player characters, and have unique interactions with them
35. As a player, I would like to experience random events that provide unique experiences each time
36. 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)
37. As a player, I would like to be able to take in-game screenshots of high quality (if time allows)
38. 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 game objects 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 allowing 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 game object in segregated JSON files. For example, one for items, another for tiles, and a third for friendly NPCs and so on. These files will be read 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 game object, 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 kilobits 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.