Team 20 Backlog

Controlled Chaos

Team Members:

Karina Abraham, Bolun Zhang, Cameron Hofbauer, Javad Baghirov, Zayden Newquist, Jack Wagner

Problem Statement:

The goal of this project is to make an entertaining game that balances a curated experience and randomly generated content. Controlled Chaos will feature the core tenets of a roguelike game, including randomly generated levels, enemy encounters, and item discovery. Our game will be different from other games by incorporating themes of randomness in the level design/artwork and by having dynamic item functions that are discovered and documented in-game by the players themselves.

Background Information:

Audience:

Typical roguelike players include people who enjoy playing games that are unpredictable in nature and include randomized features. Also, the user should enjoy problem-solving and solving puzzles. Finally, the user must also enjoy replaying games after completing them. The user base will include people of all ages.

Similar Platforms:

There are many different roguelike games which occupy various niches. The most similar game to ours is The Binding of Isaac, a 2D game where users navigate unique layouts of rooms to find items, fight enemies, and solve puzzles. Other roguelike games such as Hades, Risk of Rain 2, and Crypt of the Necrodancer take a unique spin on the genre using different gameplay loops, themes, and locations.

Limitations:

Many games require significant payment and a gaming console, such as a PlayStation or Xbox, which we will avoid by keeping the game free. Also, many modern computer games can use a lot of memory and require specialized equipment such as mice and upgraded processors. Our goal is to limit the size and processing power of this game so it can be easily played on systems without specialized hardware. We will also limit the user interface to only the screen and keyboard.

Additionally, roguelike games can be frustrating to inexperienced players due to unclear game mechanics and permanent death. By incorporating the mystery of item functionality into our gameplay loop, we ensure players are excited by this mystery and not discouraged by it. Also, by keeping our game short, we will ensure players don't get frustrated by a seemingly never-ending challenge.

Requirements:

Functional Requirements:

As a user, I would like to…

1. Move my character up, down, left, and right.
2. Attack up, down, left, and right.
3. Use ranged weapons such as a bow and arrow.
4. Use melee weapons such as a sword.
5. Use explosive weapons such as bombs.
6. Find new items such as weapons or weapon modifiers.
7. Type in the descriptions of my items as I figure them out.
8. See what items I have in an inventory and read the descriptions I gave them.
9. Damage and defeat enemies.
10. Be damaged by enemies
11. See how much health I have left.
12. See how much health enemies have left.
13. Get rewards when defeating enemies.
14. Pick up health, coins, and similar pickups.
15. Destroy game objects such as barrels.
16. Encounter friendly non-playable characters.
17. Solve puzzles such as flipping switches to open a door.
18. Pause and resume the game.
19. Quit the game.
20. Start a new game if I die.
21. Change the game's volume.
22. Change the game's resolution.
23. Enable vertical sync.
24. Save my progress on my device.
25. Enter the descriptions of enemies.

26. Name the character.
27. Change my character's appearance.
28. See animations for the character, enemies, and items.
29. Hear sounds from the character, enemies, and items.
30. Hear background music.
31. Experience different types of rooms, such as different themes.
32. Experience different effects based on the theme of the room I am in, such as fires going out in a wet area.
33. Experience different room layouts with each playthrough.
34. Experience negative or positive effects depending on the ground, such as ice making the character slide or mud making the character slower.
35. Encounter different enemy types.
36. Encounter different enemies and bosses with each playthrough.
37. Listen to different dialogue from non-playable characters with each playthrough.
38. Defeat more challenging enemies such as bosses.
39. Win the game after defeating the final boss.
40. Start a new playthrough of the game with permanent unlocks saved.
41. Access a tutorial on how the game works.
42. See a map of the level I have explored so far.
43. Write notes on the map, such as places of interest.
44. See a clock measuring how long I have played the game.
45. Get rewards for clearing levels in under a certain time.
46. Get rewards for beating a level without taking damage.
47. Permanently unlock new items to find in future playthroughs.
48. Play different characters.
49. Spend coins in a shop for new items.
50. Outsmart traps such as spikes.
51. Find secrets such as hidden rooms.
52. Choose between easy and hard areas.
53. Complete quests from non-playable characters to unlock or receive items.
54. Make my character stronger through health and damage upgrade items.
55. Give feedback to the developer to fix bugs in the game.

Nonfunctional Requirements:

Architecture and Performance:

We plan to build a game engine in Java using the LWJGL framework which gives us direct access to APIs mentioned below. This gives us more control and makes it easier to use the other APIs.

We will use OpenGL and GLFW for rendering game visuals and animations. OpenGL is a library for constructing the window used by the game; it has lower CPU overhead than other alternatives, and it is more portable. GLFW builds on OpenGL, adding code for handling user input and window management. Additionally, we will use OpenAL for sounds with in-game interactions and background music. We will compile and build the game with sbt, which comes with image reading, resizing, and writing capabilities. The game states and player positions will be saved in JSON files using GSON, a lightweight library which converts Java objects into JSON.

Our performance goal is to run the game with a frame rate of at least 30 fps. This is industry standard and should be fast enough to allow for real time reactions in combat situations without frustrating the user.

Usability:

The user interface should be easy to navigate and understand by even inexperienced players. There should be tutorials offered at the beginning of the game. These instructions should be detailed enough to give the right amount of information to the user without overwhelming him/her. Additionally, users should be able to access their progress data easily, whether that be throughout the game or at the title screen. Finally, the user should be able to interact with the game with only a keyboard, requiring no other equipment like a mouse or controller.

Deployment:

The final game will be posted on itch.io for free download and the source code will be on a private GitHub repository. This will keep the game accessible to anyone who wishes to download it.