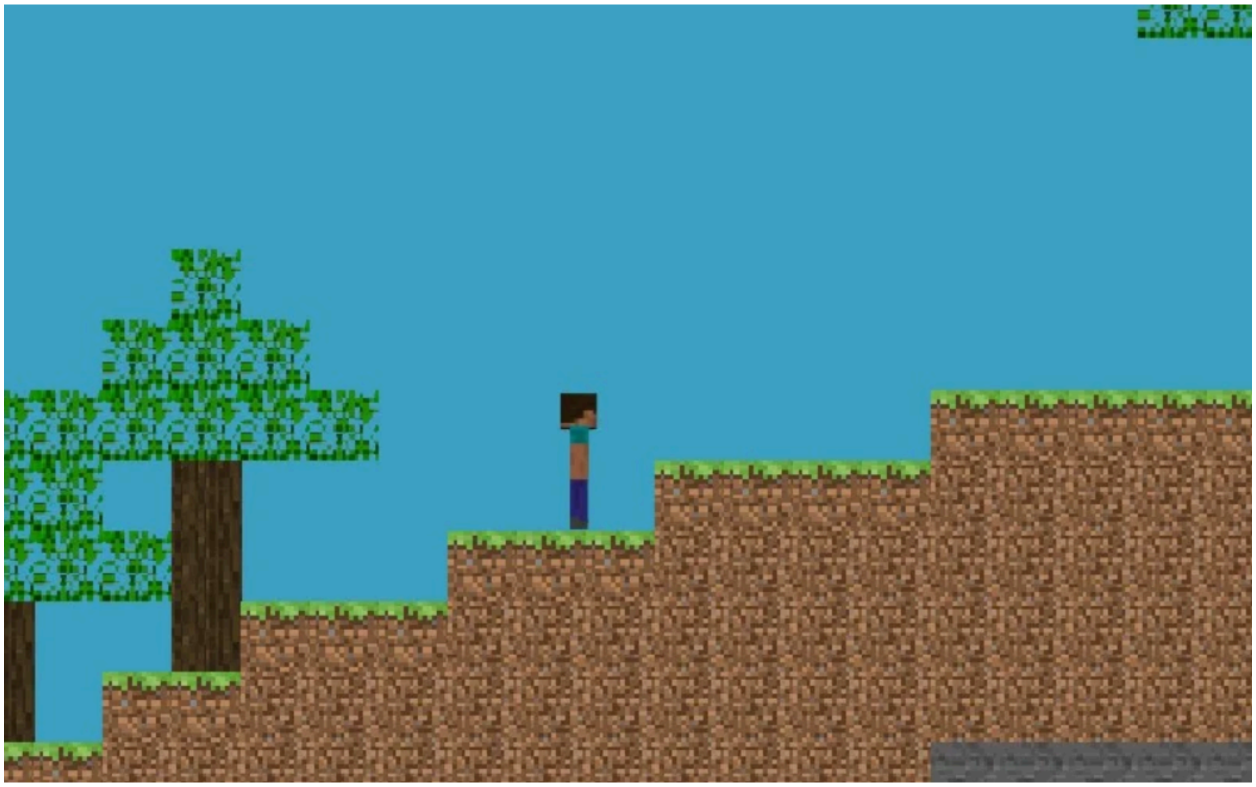
CRUx Inductions Round 3 - 2024-25 Sem 1: Aarav Gang

Duration:

14 days

Introduction:

Create a Minecraft like game in 2D with some basic mechanics implemented using PyGame.



(an image for reference)

Gameplay loop

1. Implement basic platformer controls for the player.
2. The player should be able to move around the 2D map of the game (like a platformer).
3. The player should have the ability to create and destroy blocks.

Instructions

The following features are required :

1. Create a main menu for the game containing options to:
    a. Create a new world
    b. Spawn into an already existing world
2. Terrain Generation:
    a. The world must be procedurally generated and the world must have the ability to be "saved" so that it can be entered again and played from where it was left. (You may use a noise map to generate the terrain).
    b. ✓ Like in minecraft, the world must be made of block-shaped textures.
    c. ✓ The surface of the terrain must have a grass texture, the blocks below it must be a dirt texture and the blocks below that must have a stone like texture (Similar to Minecraft)
    d. Below the main terrain, generate an underground cave system. The surface and cave should not overlap at any point; i.e the cave should never be open to the surface.
    e. Accessing the cave should be possible only via certain "wells" (Or any structure you choose). It will be only at these points that you can traverse between the cave and the world above. You can use these structures to teleport from the cave system to the surface and vice versa.
3. Destroying Blocks:
    a. The player must be able to destroy blocks within a certain range from him by clicking the block. Only blocks that are directly accessible to the player can be destroyed (Blocks that are close to the player but do not accessible directly, for example, blocks below the terrain when the player is standing above the terrain should not be able to be destroyed directly)
4. Placing Blocks:
    a. The player should be able to place any number of the block last broken by the player (Placing floating blocks should not be possible).

Brownie Points[1]

1. Create "Biomes"
    a. Different parts of the map should have different looking blocks instead of the same grass, dirt, stone pattern (For example: include blocks like sand, water etc.)
2. Place trees and any other objects randomly around the map. These objects should also be able to be broken.

[1] Brownie points are meant to be the cherry on the top, in the end the main focus is still on the points given in Instructions.

Evaluation Criteria

1. **Functionality:** Completeness and correctness of the implemented features.
2. **Code Quality:** Cleanliness and readability of the code.
3. **User Interface:** Usability and design of the UI.
4. **Performance:** Efficiency of image processing operations.

Submission Guidelines

1. Use git to record your progress throughout the induction process and use conventional commits.
2. Code quality (conciseness, modularity, error handling, etc.) and documentation (minimum required for someone to understand the project/tasks and how to run it/contribute to it) will also be evaluated.
3. Even if you can't complete your task, make sure what you submit is complete in itself (i.e., I should be able to run and test whatever you have done.)
4. Any commits after the deadline will be ignored.
5. Feel free to use the internet, but obviously, do not plagiarise.
6. Once done with your tasks, write a post on the CRUx community forum and reply to your Round 3 task email with the link to your GitHub repo. The email and post are essential; do not skip them. The post is for future applicants and the GB to check out your projects. Follow a similar pattern to those who posted from previous inductions.
7. If you have any doubts at any point or want to request some extension in the deadline for valid reasons, feel free to contact any of your task setter(s).

Task Setter: ADVAITH NEELACANTAN (8073105080)