Game Engine Documentation

Created by: S.S.M (Spazing Spades Minorities)

Purpose: To explain how our custom 2-player platformer engine works, using simple language with enough detail for everyone to get it.

1. What Our Game Is

Our game is inspired by **Ultimate Chicken Horse**. It's a fast, silly, and competitive 2-player platformer where:

- Players build the level using blocks.

- Then, they race through the level they just built.

- They try to reach the goal while avoiding traps and obstacles.

Why this matters:

This idea makes the game fun because no two rounds are the same. Players are not just playing the level they're creating the challenge for each other. This document explains how our engine works with the code we have right now.

2. Core Gameplay Loop

Build Phase

In this phase, players click a button in the side panel. They get a block that they can place anywhere in the game area. The blocks stay until removed or the level resets.

Why this step matters:

This is the creative part of the game. Players design the obstacles, platforms, and traps that they will later try to survive. It gives players control and makes each round unique.

Play Phase

Here, both players move, jump, and try to reach the goal. They can stand on the ground and the blocks they placed while avoiding pitfalls.

Why this step matters:

This is where the challenge kicks in. Players need to use skill, timing, and strategy to make it through the level they just built.

Start Screen

Before everything begins, the game shows a simple “Start” screen. Clicking it takes players to Build Mode.

Why this step matters:

It gives the game a clean start and makes it clear that players must kick off the round.

3. Current Engine Features

- **Two players (Red = WASD, Blue = Arrow Keys)**

- Movement (left, right, jump)

- Gravity

- Floor collision

- Block placement

- Side inventory panel

- Saving and loading placed blocks

- Build Mode and Play Mode

- Start Screen

Why these features matter:

These are core systems that every platformer needs. Without movement, collision, and building, the game can't function. These features form the base for future upgrades.

4. Code Structure

Global Variables

- **keyDown[]** → tracks which keys are pressed (lets both players move smoothly).

- **redPlayer, bluePlayer** → the two players (each has their own movement).

- **floor** → the ground (players need somewhere to stand).

- **onScreenBlocks** → list of all placed blocks (they must be stored for drawing and collision).

- **holdingBlock** → true when the player is holding a block (lets the game know they’re ready to place).

- **gameState** → 0 = Start, 1 = Build, 2 = Play (controls the game's current phase).

- **Screen sizes** → main area + side panel (keeps building tools separate from the play area).

Main Functions

- **settings()** → sets window size.

- **setup()** → creates players and ground.

- **draw()** → runs every frame and switches between Start, Build, and Play screens.

Why these functions matter:

They control the whole game loop. Without them, nothing would update, draw, or respond to input.

5. Build Mode

What Build Mode Draws

- The game area (left side).

- The side panel (right side).

- The inventory button.

- All placed blocks.

- The floor.

Why this matters:

It visually separates building from playing, making it easier to understand.

How Block Placement Works

- Click the green inventory button → you “pick up” a block.

- Click inside the game area → you place the block. The block is added to onScreenBlocks.

Why this matters:

This is key to the “build” part. Blocks need to be easy to place and become part of the level instantly.

Saving & Loading

- Press S → saves all block positions to a file.

- Press L → loads them back.

Why this matters:

Players can save their designs and reuse them later. This is important for testing and replaying rounds.

6. Play Mode

What Play Mode Does

- Draws the floor and all blocks.

- Updates both players.

- Applies movement, gravity, and collision.

- Draws both players.

- Shows “Press B to Build.”

Why this matters:

This is where the actual gameplay happens. Everything must update smoothly for the game to feel responsive.

Switching Modes

- Press P in Build Mode → Play Mode.

- Press B in Play Mode → Build Mode.

Why this matters:

Switching modes is essential in the “build then run” loop. Players need to switch back and forth easily.

7. Player Class

Player Variables

- **x, y** → position.

- **w, h** → size.

- **vx, vy** → velocity.

- **moveSpeed** → how fast they walk.

- **jumpPower** → how high they jump.

- **gravity** → pulls them down.

- **onGround** → true only when standing on something.

- **col** → color.

- **Key bindings** → WASD for red, arrows for blue.

Why do these matter:

These variables control how the player behaves and interacts with the world.

Player Functions

- update()

Runs every frame: reads input, applies physics, and checks collision.

Why: Keeps the player moving and reacting to the world.

- handleInput()

Moves left or right and jumps only if onGround is true.

Why: Prevents double-jumping and makes gameplay fair.

- applyPhysics()

- Adds gravity.

- Moves the player.

- **Why:** Makes movement feel natural, like in platform games.

- **verticalCollide()**

- Stops the player from falling through:

  - The floor.

  - Any placed block.

  - The bottom of the screen.

- If the player lands on something, set onGround = true.

- **Why:** Collision is super important in platformers. Without it, the game breaks.

- **drawPlayer()**

- Draws the player as a rectangle.

- **Why:** Simple shapes make testing easier before adding fancy art.

8. Platform Class

- A simple rectangle used for the floor.

- **Why:** The floor needs to be solid and easy to see.

9. Block Class

- Represents any block that the player puts down.

- Variables:

  - x, y (position)

  - w, h (width, height)

  - r, g, b (color)

- Blocks are solid, and players can stand on them.

- **Why:** Blocks are the main building tool for the game.

10. How Everything Works Together

- The game shows the world.

- The player clicks the inventory button to get a block.

- Clicking the game area puts down the block.

- Players move using WASD and arrow keys.

- Gravity pulls them down.

- Collision stops them from falling through platforms.

- Blocks become part of the level.

- Build Mode and Play Mode switch the rules.

- **Why it matters:** This flow creates the “build then run” gameplay that defines the whole game.

11. What Comes Next (Future Features)

- Movement Upgrades:

  - Wall jumping → more ways to move.

  - Coyote time → smoother jumps.

  - Dash → faster movement.

  - Double jump → more skill options.

  - **Why:** Makes movement feel more fun and advanced.

- Build Phase Upgrades:

  - More item types.
  - Rotating items.
  - Deleting items.
  - Trap behavior.
  - **Why:** Gives players more creative tools.

- **Play Phase Upgrades:**
  - Death detection.
  - Blood splats.
  - Respawn system.
  - Goal object.
  - **Why:** Makes rounds feel complete and competitive.

- **Game Systems:**
  - Scoring.
  - Rounds.
  - Random item selection.
  - Timer.
  - **Why:** Turns the engine into a full game.

12. Step-By-Step Plan

- Step 1: Core Movement & Physics (Done / In Progress)

  - Goal: Make movement feel good.

- **Why:** If movement feels bad, the whole game feels bad. Movement is key in every platform game.

- Step 2: Basic Level Building (Done / In Progress)

- Goal: Let players place blocks.

- **Why:** The idea is to “build the level, then run through it.” Without building, the game loses its fun.

- Step 3: Solid Platforms (Partially Done)

- Goal: Make placed blocks act like real ground.

- **Why**: Players need to stand on what they build for it to matter.

- Step 4: Build Phase vs Run Phase (Planned)

- Goal: Separate building from racing.

- **Why:** Makes the game clear. Players know when to build and when to run.

- Step 5: Goal Object & Scoring (Planned)

- Goal: Give players something to race toward.

- **Why:** Makes rounds meaningful by giving players a reason to move.

- Step 6: Death & Respawn (Planned)

- Goal: Make failure funny and clear.

- **Why:** Dying is part of the fun, and respawning keeps the game going.

- Step 7: Traps & Power-Ups (Future)

- Goal: Add strategy and chaos.

- **Why:** Makes each round unique. Players can set up tricky traps or give themselves boosts.

- Step 8: Polish & Extra Modes (Future)

- Goal: Make the game feel complete.

- **Why:** Themes, modes, and effects add personality. These are bonus features once the core works.