**Game Design Document**

**Jumping box Game**

**1. Game Overview**

Title: JumpCube

Genre: 3D Platformer / Infinite Jump

Platform: PC (Desktop)

Target Audience: Casual gamers, ages 8 and above, fans of simple platform mechanics

Objective:

Control a cube to jump onto platforms or navigate a static environment, demonstrating responsive physics and user input handling. The game emphasizes smooth jumping mechanics, physics feel, and minimalistic design.

**2. Game Concept**

JumpCube is a straightforward 3D game where the player controls a cube that can jump vertically. The core gameplay focuses on mastering jump timing, controlling altitude, and responding to gravity. The initial version is basic — with the primary goal being to learn physics-based movement and input handling in a 3D environment using the Ursina engine.

The player’s cube:

* Can jump when on the ground.
* Is affected by gravity.
* Interacts with the ground (static plane).
* Can be extended into more complex mechanics (e.g., moving platforms, scoring).

**3. Gameplay Mechanics**

**3.1 Controls**

* Spacebar: Initiates a jump, provided the cube is on the ground.
* Optional future controls: Move left/right (not implemented yet).

**3.2 Physics & Movement**

* The cube is affected by gravity, simulated as a constant acceleration downward.
* When the player presses 'space', an upward jump velocity is applied.
* The cube continues to rise until gravity decelerates it, then it falls back down.
* Ground collision detection prevents falling through the ground plane.

**3.3 Game Loop & Update**

* On each frame, gravity influences vertical velocity.
* The position is updated based on the current velocity.
* When hitting the ground, velocity resets, allowing for subsequent jumps.
* No obstacles or scoring are present yet, but mechanisms can be added later.

**4. Visual Design**

**4.1 Environment**

* Plain ground plane: gray cube scaled to serve as ground, positioned at y = -2.
* Player cube: orange, 1x1x1 in size.
* Background: default white/gray from Ursina (can be expanded).

**4.2 Camera**

* Default camera, facing the scene from a distance.
* Future improvements could include dynamic camera or different angles.

**4.3 Aesthetics**

* Minimalistic 3D scene.
* Simple color palette for clarity and performance.

**5. Implementation Details**

**5.1 Entities**

* Ground: static cube scaled as a flat plane.
* Player: controlled entity with physics for jump.

**5.2 Physics Simulation**

* Gravity constant = -12 units/sec².
* Vertical velocity is updated in each frame (velocity\_y += gravity \* dt).
* Player's position modified accordingly (player.y += velocity\_y \* dt).

**5.3 Collision Detection**

* Checks whether the player has hit the ground (player.y <= 0).
* Resets position and velocity when landing.

**5.4 Input Handling**

* Detects 'space' key press.
* Initiates jump only if on the ground.

**5.5 Code Structure**

* The core logic is within the update() function.
* Input is handled via input().

**6. Future Enhancements & Features**

* Add obstacles (walls, pits).
* Introduce scoring based on jump height or time survived.
* Implement left/right movement.
* Add power-ups or collectibles.
* Create multiple levels or environments.
* Implement audio effects and visual feedback (jump sound, landing effects).
* Add a UI with start screen, instructions, and game over screen.
* Multiplayer or online leaderboards.

**7. Technical Considerations**

* Performance: Ursina handles rendering efficiently, suitable for simple 3D prototypes.
* Portability: Works on Windows, Linux, and Mac with Python installed.
* Expandability: Modular code, easy to add features or modify controls.

**8. Summary & Conclusion**

This JumpCube project provides a foundation for physics-based 3D movement in Ursina. It is designed for simplicity and clarity, making it accessible for beginners while allowing room for future complexity.