

ShatterSprint - Documentation

Game Controls:

The controls in ShatterSprint are pretty simple. There is a constant movement speed throughout the level and you can double it by holding **W** and halve it by holding **S**. You can also increase or decrease the constant speed by **scrolling** if it's too fast or too slow for your current situation. If you need to strafe to the side, you can do so by pressing **A** and **D**. You can help this movement by looking slightly to the left or right by moving your mouse. How far you can look to your left or right is limited however. By pressing **Space**, you can jump over gaps and even glide if the according powerup is active.

For the prototype, you can also press **2** to make all tiles fall onto an invisible plane, where they will despawn in the final game and increase your score. Pressing this button is probably not the best idea now.

Furthermore, the game can be paused with **Esc** and can be closed from the menu by pressing **Q**.

State of Development:

The game is in a playable state and the goal is to make as many tiles as possible fall by touching them. Dont fall into the gaps, or its game over. The player can collect powerups to glide and earn more points and also has to deal with blocks potentially falling right under their feet.

Implemented Requirements:

In the full version the following requirements from the “*Gameplay*” section are implemented:

- 3D Geometry (Powerup buildings and powerups)
- 60 FPS and Framerate Independence
- Win/Lose condition (Falling down means game over)
- Intuitive controls (wasd/space)
- Intuitive camera (mouse controlled camera)
- Illumination model (directional lighting)
- Textures (multiple textures for models, decals and particles)
- Moving objects (rotating powerups)
- Documentation (Its this right here!)
- Adjustable Parameters (Resolution/Fullscreen can be changed in the config file)

The following requirements from the “*Advanced gameplay*” section are implemented:

- Collision Detection (Lose ground and you fall, collision for the powerup buildings)
- Advanced physics (Triggers for falling cubes, reset of the players jump)
- HUD (Point counter and menu hint)

The following requirements from the “*Effects*” section are implemented:

Advanced Modeling:

- CPU particle system (Effects around powerups)

Texturing:

- Procedural texture (The temple material texture)

Post processing:

- Motion blur (Should be noticeable by slightly increasing movement speed or shaking the camera around)

External Resources:

- For Physics Bullet is used in version 3.25, specifically BulletCollision, BulletDynamics and LinearMath.
- For object loading Assimp is used in version 5.4.0. Assimp is compiled as a shared library so corresponding .dll files should exist in the folder of the executable.
- The file “FirstPersonCamera.h” is heavily inspired by <https://learnopengl.com/Getting-started/Camera>
- The file “Model.h” is heavily inspired by <https://learnopengl.com/Model-Loading/Model>
- The particle system was extended from <http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/>
Added several options to configure a particle spawner
- Text rendering was inspired by <https://lwjglgamedev.gitbooks.io/3d-game-development-with-lwjgl/content/chapter12/chapter12.html>
- The implementations for perlin noise were inspired by <https://adrianb.io/2014/08/09/perlinnoise.html>