

## DEPENDENCIES & THIRD-PARTY LIBRARIES

| Library                          | Version  | Purpose                                                       | Via                                |
|----------------------------------|----------|---------------------------------------------------------------|------------------------------------|
| GLFW                             | 3.x      | Window creation, OpenGL context, input events                 | Vcpkg                              |
| GLAD                             | -        | OpenGL 3.3 function loader                                    | vcpkg                              |
| GLM                              | 0.9.9+   | Math: vec3, mat4, quat, perspective, lookAt etc.              | vcpkg                              |
| stb_image.h<br>stb_image_write.h | 2.x      | PNG/JPG texture loading/writing (Sean Barrett, public domain) | Single-header, embedded in project |
| OpenGL                           | 3.3 Core | GPU rendering API                                             | System / driver                    |
| Bullet3                          |          | Collision detection                                           | vcpkg                              |

Sean T. Barrett — Public Domain / MIT License

stb\_image reference: [https://github.com/nothings/stb/blob/master/stb\\_image.h](https://github.com/nothings/stb/blob/master/stb_image.h)  
[https://github.com/nothings/stb/blob/master/stb\\_image\\_write.h](https://github.com/nothings/stb/blob/master/stb_image_write.h)

GLM documentation: <https://glm.g-truc.net/0.9.9/index.html>  
<https://github.com/g-truc/glm>

GLFW documentation: <https://www.glfw.org/docs/latest/>

GLAD generator: <https://glad.dav1d.de/>

Bullet3 <https://github.com/bulletphysics/bullet3>

## Controls

### In Game in general

| Input                     | Action                                       |
|---------------------------|----------------------------------------------|
| W                         | Move ball forward (camera-relative)          |
| A                         | Move ball left (camera-relative)             |
| S                         | Move ball backward (camera-relative)         |
| D                         | Move ball right (camera-relative)            |
| Space                     | Jump (only when ball is touching the ground) |
| Right Mouse Button (hold) | Orbit camera around ball                     |
| Q                         | Zoom camera in                               |
| E                         | Zoom camera out                              |
| M                         | Return to Main Menu                          |
| ESC (global)              | Quit the application                         |
| N                         | Enable/Disable Normal maps                   |

### IN GAME - SCREEN LEVEL COMPLETE /SCREEN GAME OVER

| Input | Action                                      |
|-------|---------------------------------------------|
| SPACE | Proceed to next level / Retry current level |
| M     | Return to Main Menu                         |

## DEBUG CONTROLS (GLOBAL)

| Input | Action                                               |
|-------|------------------------------------------------------|
| F8    | Toggle free-fly Debug Camera (overrides ball camera) |
| F9    | Toggle FPS counter overlay (bottom-right corner)     |
| F10   | Toggle wire frame render mode                        |
| F11   | Toggle back-face culling                             |

## FREE-FLY DEBUG CAMERA (F8 ACTIVE)

| Input                    | Action                                   |
|--------------------------|------------------------------------------|
| W / A / S / D            | Fly forward / left / backward / right    |
| Q                        | Fly up                                   |
| E                        | Fly down                                 |
| Left Mouse Button (drag) | Look around (free-look)                  |
| Shift (hold)             | Turbo speed                              |
| F8                       | Exit debug camera, return to ball camera |

## VISUAL EFFECTS

### PROCEDURAL TEXTURES

All textures are generated at startup by the class „ProceduralTexture“ and uploaded to the GPU once.

| Texture      | Noise type              | Appearance                                                         |
|--------------|-------------------------|--------------------------------------------------------------------|
| WOOD         | Perlin fBm + turbulence | Concentric ring grain pattern with lacquer sheen.                  |
| STONE        | Worley + Perlin         | Irregular cracked-stone cells with dark joints.                    |
| ICE          | Perlin fBm              | Pale blue-white smooth ice surface.                                |
| MARBLE       | Perlin turbulence       | White and gray veined marble.                                      |
| DIRT         | Worley + fBm            | Warm brown multi-octave fBm with pebble hints from high-freq layer |
| COBBLESTONE  | Worley + fBm            | Thin dark region near cell boundaries                              |
| CHECKERBOARD | Analytical              | High-contrast black and white checker grid                         |

Each texture also has a matching specular map (R channel) that controls how shiny each material appears under the directional sun light.

### SHADOW MAPPING WITH PCF

A directional shadow map is rendered every frame from the sun's point of view. Geometry is re-rendered into a  $4096 \times 4096$  depth texture. The Phong shader then compares each fragment's depth against this map to determine shadowing. PCF Kernel with a Percentage Close Filtering (soft shadow edges)

## ***SPECULAR MAPS***

Every procedural texture has a companion specular map. The Phong fragment shader samples the R channel of this map (uSpecularTex) to modulate the Blinn-Phong highlight intensity on a per-textel basis.

| Aspect       | Detail                                                                                  |
|--------------|-----------------------------------------------------------------------------------------|
| Map format   | Single-channel (R); white = fully specular, black = fully matte.                        |
| Wood example | High specularity along ring edges (lacquered finish), low on flat grain.                |
| Ice example  | Uniformly high — smooth ice reflects strongly everywhere.                               |
| Fallback     | If no specular map is provided (uUseSpecTex = false), a constant value of 0.45 is used. |

## ***NORMAL MAPS***

Every procedural texture has a companion normal map, generated from the same underlying height function as the diffuse texture. The Phong fragment shader samples this map (unit 3, uNormalMap) to replace the interpolated geometry normal with a per-textel surface normal, giving the illusion of fine surface relief without additional geometry.

| Aspect     | Detail                                                                                                          |
|------------|-----------------------------------------------------------------------------------------------------------------|
| Map format | RGB; flat surface = (128, 128, 255); XY channels encode gradient, Z encodes height                              |
| Toggle     | N key (keyNormalMap) — switches between mapped and geometry normal                                              |
| Fallback   | When disabled or no texture is bound (uUseNormalMap = false), the unmodified interpolated vertex normal is used |

## ***LENS FLARE***

The sun produces a multi-element lens flare that fades in and out as it enters or leaves the camera's field of view and as it is occluded by geometry.

| Aspect         | Detail                                                                                                                                                           |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Occlusion test | Every frame the sun is projected to screen space. glReadPixels reads the depth buffer at that pixel; if the sun's depth > scene depth it is considered occluded. |
| Fade           | Visibility smoothly interpolates to 1 (visible) at 7x/s and to 0 (hidden) at 3x/s, giving a gentle fade both ways.                                               |
| Blending       | Additive blending (GL_SRC_ALPHA, GL_ONE) so overlapping elements brighten each other, simulating camera lens scattering.                                         |
| Sun tint       | Flare color is derived from the level's sun color, so warm suns produce warm flares and cool suns produce bluish flares.                                         |

## ***CPU Particle System***

Picking up coins creates 40 particles. These disappear within 1 second.

## Features

| Tile types | Detail                                           |
|------------|--------------------------------------------------|
| Normal     |                                                  |
| Ice        | Reduces friction of the ball (slides)            |
| Bouncy     | Normal landing changed to launch upward          |
| Crumble    | Tile vanishes if ball stands on it for some time |

| Power up Types | Detail                         |
|----------------|--------------------------------|
| Speed Boost    | Increase the speed of the ball |
| Anti Gravity   | Reduce Gravity on jump/fall    |
| Shield         | Survive one fall               |

| Settings Types | Detail                                                             |
|----------------|--------------------------------------------------------------------|
| Resolution     | Can be changed to: (800x600,1280x720,1600x900,1920x1080,2560x1440) |
| Full screen    | Yes/No (change window to full screen if yes)                       |
| FPS Cap        | Can be changed to:(60,100,120,200)                                 |
| Y-axis inv.    | if on invert the y-coordinates in game (change camera position)    |

### ***Level timer***

Each level has a specific time to complete. When time runs out, the game is over.  
Change the sun position with level timer (y-coordinate).

### ***Lives***

At the begin of the game the player gets 3 lives. If the player falls to death ( $y < -50$ ) one life is lost.  
If all lives are lost it is a Game Over.

### ***Movable objects***

Movable tiles that can move along one axis (x,y,z).

### ***Collision Detection (Bullet)***

### ***Coin collection***

The exit tile (green colored tile) will work to complete the level after all coins of the level are collected.

### ***Blender objects***

Level 1 2 3 the Coins are loaded from Coin.obj

Level 1 and 3 the Power ups (Speed Boost and Anti Gravity) are loaded from Gravity.obj and SpeedMod.obj