

Mudslide Madness

Starting the Game

When the game starts, it loads configuration from "assets/settings/settings.ini"
Resolution, fullscreen mode and camera properties can be adjusted.

Current Controls

W A S D Control Car (or debug camera)

Left Mouse - Change look direction of the debug camera

P - Switch between Follow and Debug Camera

R - Reset Level

H - Toggle HUD

F3 - Toggle Cursor lock

ESC - Exit Game

F1 - Change Polygon Mode (Full | Wireframe)

F2 - Change Polygon Culling (Remove nothing | Remove backfacing triangles |
Remove frontfacing triangles)

N - Toggle normals rendering

T - Toggle texcoord rendering

Gameplay Features

3D Geometry

The player controlled car and the track (.obj) are loaded using the library ASSIMP

Playable

A car can be driven around the race track.

Framerate Independence and min. 60FPS

All movement and time related data is multiplied by the last frame time to achieve framerate independence. Performance was tested using [mangohud](#). It always exceeded 60 FPS.

Win/Loose Condition

The goal is to drive the car around the track as fast as possible. There are several hidden checkpoints that have to be passed in order for a lap time to count.

The car can get damaged when driving into obstacles reducing it's health. When health reaches 0 the car loses its power and the level has to be reset to continue. Also missing checkpoints results in an invalid lap.

Controls

Continuous input is implemented using polling (eg. **W** **A** **S** **D**) and single events (e.g **P**) use callbacks. see Controls.h and Controls.cpp

Camera

For our game, a fixed camera is used (FollowCamera Class). Pressing **P** toggles the debug camera, which can be moved around freely using **W** **A** **S** **D** and **Left Mouse**

Illumination Model

Currently there is a fixed directional light (sun). All visible objects (except particles) have a material and use the phong lighting model.

Textures

The game supports textures. (Currently only the trees have a texture, everything else are materials / vertex colors). Textures are correctly mipmapped and use trilinear filtering. See DdsLoader.h and DdsLoader.c for the implementation.

Moving objects

Physically simulated moving car.

Adjustable Parameters

All adjustable parameters are located in "assets/settings/settings.ini"

- Resolution
- Fullscreen mode
- Background color
- Camera fov
- Camera near and far plane

Optional Gameplay Features

Collision Detection (Basic Physics)

The library [JoltPhysics](#) is used to simulate collisions.

Advanced Physics

The library [JoltPhysics](#) is used to simulate a controllable car. When the car collides into static objects, it gets damaged. After a few hits, the engine starts emitting a white smoke. The more damaged the car is, the more smoke is emitted and the blacker the smoke is.

Heads-Up Display

HUD to display (best) lap time and game over screen. Can be toggled using H

Effects

Shadow Map with PCF

A separate render pass is used to render the scene from the light's perspective. The depth attachment of this pass is then used in the normal render pass to be able to evaluate if the pixel is in a shadow or not. Shadow acne is dealt with the hardware bias functionality provided by vulkan and an additional (very small) bias in the fragment shader. Peter Panning is no problem, as there are no thin objects / the shadow map resolution is high enough. (Hardware) PCF is enabled for the shadow map. Only the provided materials from the TUWEL course were used as reference:

- [Shadow Mapping Tut from TUWIEN](#)
- [Sasha Willems' Vulkan shadow mapping example](#)

GPU Particle System using compute shader

See Particles.h and Particles.cpp.

The Particle system manages its particles in a compute shader. The particles have a position, velocity, color and time-to-live. The particle system was implemented using the [Tutorial from TUWien](#) as reference.

The particles can be seen when driving the car.

Hierarchical Animation

The wheels of the car are separate meshes that roll / turn with respect to the car's speed and steering

Video Texture

A 3 second 30fps portion of the MudslideMadness video plays on one of the screens of the tv stand on the map Can best be seen when using the debug camera (P)

Libraries

[ASSIMP](#) is used as a object loader. The logic for object loading can be found in "loadModel()" in "Geometry.cpp"

[JoltPhysics](#) is the physics engine of the game. Implementation can be found in the src/Jolt folder