

Orrery Drift

PTVC Submission 2

Group Name: Orrery Drift

Students:

- Peter Kabelik: 01125096
- Diego Garcia Straub, 12428637

Genre: Kart racing game

Gameplay Summary:

You Press enter and see the Planet you're going to race around spinning. (Check out the Atmosphere!) Press enter again and after the countdown the race starts. Follow the arrows through the racetrack, try to go as fast as possible, and don't forget to drift. After 3 Laps the Race ends, and you see your score in the Trophy Scene. Positions one through three get gold, silver or bronze. What's your best time?

Feature	Description
3D Geometry (6 Points)	The game has 3d meshes, loaded from .obj with a custom importer. The meshes were modeled and textured in Blender.
Playable (3 Points)	The game is playable. Controls and gameplay are further described below
Min. 60 FPS and Framerate Independence (3 Points)	The game is framerate independent. The delta time is used for all physics calculations, as well as camera and movement.
Win/Lose Condition (3 Points)	At the end of the race, you're placed in Position 1-6. For the first 3 places you get a Trophy.
Intuitive Controls (2 Points)	Game can be controlled with Keyboard and mouse or controller. Check below for Controls
Intuitive Camera (2 Points)	On keyboard the camera can be moved with the mouse.
Illumination Model (2 Points)	Every Model has a Material (PBR), there are multiple light sources (1 directional). Every Model has normals (some have normal Maps)
Textures (2 Points)	All Objects have mipmapped textures.
Moving Objects (2 Points)	The Player vehicle can move, as well as some boxes in the scene.
Documentation (1 Point)	This is the Documentation
Adjustable Parameters (1 Point)	Resolution and Fullscreen (as well as v-sync) can be adjusted in settings.txt config file.
Advanced Gameplay (5 Points)	The Gameplay is advanced (whatever that means).

Collision Detection (Basic Physics) (6 Points)	The Bullet Engine provides the collision detection we used in the entire engine
Advanced Physics (4 Points)	We have vehicles and callback with our checkpoint system (collision with checkpoints set new respawn point)
Heads-Up Display (4 Points)	We have a speedometer and information on lap-time and current lap

Effect	Description
Shadow Map with PCF (16 Points)	We use a shadow map (render from the sun's perspective). With PCF for smoother shadows
CPU Particle System (8 Points)	When drifting the boat shoots out mist particles, which are computed on the CPU and rendered using instanced quads
Procedural Texture (8 Points)	The Ocean is textured at runtime in the fragment shader using information from the depth Buffer and moving Normal maps
Specular Map (4 Points)	We have Roughness and Metallic maps (as well as Ao maps) for PBR. They are combined in one image in ARM format.
Triplanar Texturing (6 Points)	The ocean is normal mapped using triplanar mapping to archive normals for a mesh without uv maps
Environment Map (8 Points)	The Game has a skybox (night sky) implemented as an HDR-cube map.
Image Based Lighting (10 Points)	We have IBL implemented in form of spherical Harmonics for diffuse and pre-filtered env-maps for specular. Can be best viewed in Trophy Scene
Simple Normal Mapping (4 Points)	We have simple Normal mapping (eg. Ocean) (can be toggled with F12). Not all meshes have normal Maps.
Physically Based Shading (6 Points)	We have PBR implement using the Cook-Torrance model for the entire scene.
Bloom/Glow (8 Points)	We have implemented Bloom/glow using a blur over bright spots. Can be noted in Glowing Objects (Arrow-signs, certain house windows)
Contours via Edge Detection (12 Points)	We have edge detection using both information from normals and depth. We use these to create Contours around the edges.
Ambient Occlusion (12 Points)	We have implemented SSAO in low res and up sampled using a (Joint) bilateral filter.

Special Features:

Atmosphere:

We have implemented atmospheric rendering using a custom approach without raymarching. (Analytical solution to the integral, assuming a linear falloff, of atmospheric density). Only Rayleigh scattering.

Sound:

We have a backing track. And an Intro Countdown to the race

Rendering Type:

We render the scene in passes using deferred shading. Check more detailed second documentation for more Info

Other Racers:

We have five other races. Your previous race data is one of the racers if available, else static prerecorded race data is used. These racers don't have collisions; their positions and rotations are interpolated.

Libraries:

Vulkan 1.3.290

GLFW 3.3.9

stb_image.h 2.30

stb_truetype.h 1.26

stb_image_write.h 1.16

glm 1.0.3

Bullet 3.27

fastgltf 0.9.0

miniaudio 0.11.25

Keyboard Controls	Controller Controls
WASD: car Movement Mouse: Camera rotation Shift: Drift R: Respawn Enter: Continue	Right Trigger: Accelerate Left Trigger: Brake / Reverse Left Joystick: Steer Right Bumper: Drift A: Respawn / Continue

Extra Controls:

F1- Show FPS F2- Toggle Debug Info F3-F4 Adjust Song Volume F6 Toggle Free Cam F7 Toggle Collision hitbox F8-F9 Scroll through Debug Views (Loops) → F10 Show Light Bound Spheres F11 Switch Tone-mapping / Color correction F12 Toggle Normals Free Cam Controls: IJKL: Movement U/O: UP/Down Arrow Up/ Arrow Down: Movement Speed	0: Standard 1: Point Lights + Ambient Light 2: Ambient Light 3: Point Lights 4: Directional Sun Light 5: Shadows 6: Roughness Map 7: Metallic Map 8: Emissive Map 9: Ao Map 10: SSAO 11: Shadow Map depth 12: Normals 13 Linear Depth 14: Albedo 15: Contours (Edge Detection)
---	---

Extra Credits:

Countdown sound from:

<https://opengameart.org/content/race-start-countdown> (CC0)

Music from:

[https://commons.wikimedia.org/wiki/File:Smetana_-_M%C3%A1_Vlast_-_Vltava_\(Musopen_Symphony\).flac](https://commons.wikimedia.org/wiki/File:Smetana_-_M%C3%A1_Vlast_-_Vltava_(Musopen_Symphony).flac)

Trophies and HDRIs from:

<https://polyhaven.com/hdris>

Space HDRI from:

<https://www.spacespheremaps.com>

Tutorials and guides used:

SSAO up sample:

<https://bartwronski.com/2019/09/22/local-linear-models-guided-filter/>

PBR rendering:

learnopengl.com/PBR/Lighting

graphicrants.blogspot.com/2013/08/specular-brdf-reference.html

Ambient Light

learnopengl.com/PBR/IBL/Diffuse-irradiance

learnopengl.com/PBR/IBL/Specular-IBL

cseweb.ucsd.edu/~ravir/papers/envmap/envmap.pdf

Shadow map:

learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping

Atmosphere and Ocean:

youtu.be/lctXaT9pxA0

Bloom:

<https://learnopengl.com/Advanced-Lighting/Bloom>

Tone mapping:

<https://learnopengl.com/Advanced-Lighting/HDR>

Contours:

https://tuwel.tuwien.ac.at/pluginfile.php/4783557/mod_page/content/133/CelShading_SS19.pdf