Race Out of Space 3202

Julian Zeilinger[1] and Dominik Forkert[2]

[1]**e12122540@student.tuwien.ac.at**
[2]**e0825483@student.tuwien.ac.at**

BRIEF DESCRIPTION OF IMPLEMENTATION

In this game you navigate a spaceship through a tubular environment filled with obstacles. Interaction between ingame objects are handled by the PhysX library with an object damage system based on event callbacks .
Each ingame object is registered with the `EntityManager` class which dynamically handles object creation/destruction as well as controller input, buffer updates, and draw calls.
The game employs a custom multipass rendering pipeline to handle shadowmapping and bloom effects. Compatibility of this pipeline with the ECG framework is ensured via the class `ECGInterals` which exposes internal ECG variables otherwise hidden.
Additional implementation details for each gameplay feature and effect are provided in the sections below.

ADDITIONAL LIBRARIES

In addition to the ECG framework, the following libraries are used:

- *TinyOBJ* [11]
- *Dear ImGui* [4]
- *PhysX* [8]

CONTROLS

The table below describes the mouse and keyboard controls currently implemented in the game.

Key	Action
`Mouse left-click`	Drag camera
`Mouse right-click`	Strafe camera
`Mouse scroll`	Zoom camera
`A, D`	Strafing in the horizontal plane of the spaceship
`W, S`	Accelerating / decelerating
`Q,E`	Rolling along the forward axis
`Space`	Braking
`Esc`	Quitting the game
`F1`	Toggle wireframe mode
`F2`	Toggle backface culling mode
`F3`	Toggle camera lock
`F4`	Toggle HUD
`F5`	Toggle Tessellation mode for player spaceship
`F6`	Toggle Tessellation level

IMPLEMENTED FEATURES

Compulsory Features

Points	Item	Description
6	*3D Geometry*	All models in-game (such as player ship or containers) are imported via the *TinyOBJ* library [11].
3	*Playable*	Control the player spaceship using the keyboard controls described in the section above.
3	*Advanced Gameplay*	Evasion of static and dynamic obstacles.
3	*Min. 60 FPS and Framerate Independence*	Framerate independent game-speed is implemented via `glfwGetTime()` in the class `DeltaTime`; to deactivate *VSync* change `present_mode_immediate` to `true` and adjust `refresh_rate` accordingly in `/bin/assets/settings.ini`.
3	*Win/Lose Condition*	*Win:* reach the end of the level. *Lose:* player health drops to 0.
3	*Intuitive Controls*	Genre-typical keyboard controls as described in the section above.
3	*Intuitive Camera*	Third-person camera which can be adjusted via mouse as described in the section above.
2	*Illumination Model*	Currently 1 point light and 1 global light; per-object materials are defined in the `entityUniqueGraphics` struct; normal vectors are imported via OBJ.
2	*Textures*	Currently all models imported via *TinyOBJ* have textures attached.
2	*Moving Objects*	Player ship and containers are dynamic *PhysX* objects.
1	*Documentation*	This PDF file.
1	*Adjustable Parameters*	Can be adjusted via the `Settings.ini` file in the game directory.

Optional Gameplay Features

Points	Item	Description
4	*Collision Detection (Basic Physics)*	Simulation of dynamic and static objects based on *PhysX* [8]; collision events with damage model based on simulated collision forces.
6	*Advanced Physics*	See above.
0	*Heads-Up Display*	Implementation based on the *Dear ImGui* library [4] (no points).

Effects

Points	Item	Description
16	*Shadowmap with PCF*	implemented via a separate render sub-pass based on the tutorials [9], [10]; artifacts (*shadow acne*, *Peter panning*) are prevented via variable depth bias and focusing the center of the shadowmap on the player spaceship.
20	*Subdivision surfaces*	Implemented via tessellation shaders based on algorithm [17] and tutorial [5] for the player spaceship mesh.
8	*Vertex Shader Animation*	Force fields at the level boundaries based on time- and space-dependent vertex offsets using a periodic sine function (no particular tutorial used).
8	*Environment Map*	Implemented as part of the *PBR* pipeline using both, an irradiance cube-map for diffuse reflections and a pre-filter cube-map for specular reflections; based on tutorials [6], [7].
16	*Physically-based Shading*	Texture-based PBR according to tutorial [7].
8	*Bloom*	Gaussian Bloom effect using special *emit* textures per model in an additional off-screen render sub-pass; based on tutorial [3].

ACKNOWLEDGMENTS

The following models are used under the *Attribution 4.0 International* licence [1]:

- *Space Ship* [18]: unmodified
- *Space Station Modules* [15]: only parts of the model scene are shown in-game
- *Post Apocalyptic Shipping Container* [16]: unmodified

The following textures are used under the *Attribution-ShareAlike 3.0 IGO* licence [2]:

- *The colour of the sky from Gaia's Early Data Release 3* environment map [14]: original image converted to cubemap

REFERENCES

[1] Attribution 4.0 International. https://creativecommons.org/licenses/by/4.0/legalcode.

[2] Attribution-ShareAlike 3.0 IGO. https://creativecommons.org/licenses/by-sa/3.0/igo/.

[3] Bloom (Sascha Willems Vulkan examples). https://github.com/SaschaWillems/Vulkan/tree/master/examples/bloom.

[4] Dear ImGui. https://github.com/ocornut/imgui/.

[5] Model tessellation (Sascha Willems Vulkan examples). https://github.com/SaschaWillems/Vulkan/tree/master/examples/tessellation.

[6] PBR image based lighting (Sascha Willems Vulkan examples). https://github.com/SaschaWillems/Vulkan/tree/master/examples/pbribl.

[7] PBR (Learn OpenGL). https://learnopengl.com/PBR/Theory.

[8] PhysX. https://github.com/NVIDIAGameWorks/PhysX/.

[9] Shadow Mapping (Learn OpenGL). https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping.

[10] Shadow mapping (Sascha Willems Vulkan examples). https://github.com/SaschaWillems/Vulkan/tree/master/examples/shadowmapping.

[11] TinyOBJ library. https://github.com/tinyobjloader/tinyobjloader/.

[12] Vulkan-Dev. https://vkguide.dev.

[13] Vulkan-Tutorial. https://vulkan-tutorial.com.

[14] ESA/Gaia/DPAC. *The colour of the sky from Gaia's Early Data Release 3*. https://www.esa.int/ESA_Multimedia/Images/2020/12/The_colour_of_the_sky_from_Gaia_s_Early_Data_Release_32. License: CC BY-SA 3.0 IGO.

[15] re1monsen. *Space Station Modules* model. https://skfb.ly/oBurs. License: CC Attribution.

[16] S. Virmani. *Post Apocalyptic Shipping Container* model. https://skfb.ly/6vqYF. License: CC Attribution.

[17] A. Vlachos, J. Peters, C. Boyd, and J. L. Mitchell. Curved PN triangles. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 159–166, 2001.

[18] yanix. *Space Ship* player model. https://skfb.ly/LzKz. License: CC Attribution.