Balloon Race 3D

BalloonRace3D is a multiplayer air race with ballons. The players have to fly a given amount of rounds in a course of rings. The rings have to be taken in a given order. Both players start at the same distance from the first ring and the player who passes the last ring first wins.

Controls

The ballons have an airscrew in the back of the basket so they can move forwards/backwards and are steered with a rudder. They move up and down as usual by heating or letting out air.

NOTE: A detailed description for controlling the Actors is in the Design Document.

Key Player 1/2 (with XBOX 360 Gamepad)	**Effect**
Joystick left	Balloon Accelerate/Decelerate/Pan left/Pan right
Joystick right	Balloon Rotate view around balloon
A BUTTON (Gamepad)	Balloon Fire rocket
RT	Balloon Go up
LT	Balloon Go down
ESC	Quit game
F1	Help (in Consol)
F2	Frame Time on/off
F3	Toggle wireframe
F4	Textur-Sampling-Quality*: Nearest Neighbor/Bilinear
F5	Mip Mapping-Quality*: Off/Nearest Neighbor/Linear
F6	HDR on/of
F7	-
F8	Viewfrustum-Culling on/off
F9	Transparency on/off

*changes can be best seen at the cube map

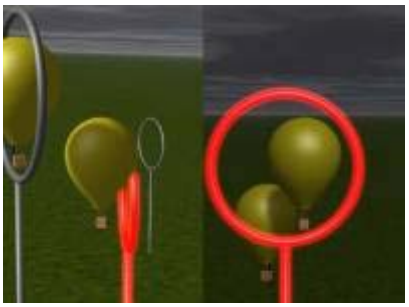
Features

- fun game :)
- 2 player split-screen multiplayer
- Collision Detection
- GPU-Particle System of the rockets; Balloon can shoot
- Changing Speed of the actor (slows down if hit by a rocket; accelerate if payer “collects” speed cube)
- Passed detection for rings
- Displaying each player (seperately) which ring it should fly through next (next ring glows)

Design Document

Design Document

Screenshots



Balloon Race 3D Design Document

Description

Balloon Race 3D is a multiplayer racing game with hot-air balloons through a parkour of rings.

Gameplay

Two players are compeating against each other. Through a parkour of freestanding rings (like the rings in Quidditch) every player navigates a hot-air balloon. The balloons have a propeller and a rudder at the back of the cabbin to move forward and to steer. To get throught the rings, which are placed in different heights, the balloons also have to increase/descrease their flying height. The balloon which reaches the finish ring (first ring for the second time; one round) first is the winner. The winner is printed out in the Console. Just one balloon fits through the rings at the same time. To get ahead the other balloon, there are several posibilities to obtain an advantage. First the other balloon can be shot with a rocket, so that the other balloon gets damaged and slowed down. There are collectable goodies, which make your actor faster if they get collected. There is one game maps.

Controls

Key Player 1	Key Player 2 (with Numpad)	Key Player 1/2 (with Gamepad)	Effect
W	8	Joystick (right) up	Accelerate
S	5	Joystick (right) down	Decelerate/Backwards
A	4	Joystick (right) left	Pan left
D	6	Joystick (right) right	Pan right
2	Arrow up	RT	Go up
1	Arrow down	LT	Go down
SPACE	+	A	Shoot Rocket
ESC			Quit game
F2			Change controls
F3			Toggle wireframe
F4			Textur-Sampling-Quality: Nearest Neighbor/Bilinear
F5			Mip Mapping-Quality: Off/Nearest Neighbor/Linear
F6			HDR on/of
F7			-
F8			Viewfrustum-Culling on/off (shows in console how many objects are culled)
F9			Transparency on/off

Technical Details

The race takes place in an arena which is as high as the balloons can fly.

Types of objects in the game

All objects in the scene are textured and illuminated. The objects are:

- The balloons – Animated character mesh
- Arena – box-like container of the game
- Rings – static mesh
- Rockets - animated mesh - with smoke as particles
- Extra Speed Goodies - a textured cube and transparency*

All models were created in Blender (and exported as .dea- files): https://www.blender.org [https://www.blender.org]

*transparency at the “Extra Speed Goodies” can be best seen if you manoeuvre the balloon below the Cube and turn the camera. If transparency is turend on you can see the balloon through the cube

Illumination

Als Beleuchtungsmodell wird Blinn-Phong verwendet. Dazu werden die Oberflächennormalen n interpoliert und basierend auf dem Winkel zwischen n und dem Lichtstrahl l die diffuse Helligkeit und mittels dem Winkel zwischen n und dem Reflektionsstrahl r der Glanzpunkt (speculare Helligkeit) berechnet.

Das Beleuchtungsmodell wird auf alle Objekte angewendet. Die einzige Ausnahme bildet hier die Arena/der Cube. Auf diese Oberfläche wird nur “texture color” verwendet.

Texture

- Arena: gras.jpg (Source: http://www.f-lohmueller.de/pov_tut/backgrnd/p_sky9f.htm [http://www.f-lohmueller.de/pov_tut/backgrnd/p_sky9f.html])
- Extra Speed Goodies: forward.jpg

Camera

The screen is split between the players. Each player has his/her own subscreen and camera. The cameras automatically follow the balloons from several meters distance. The players can control the cameras either through the left Joystick of the gamepad or by using keys on the keyboard (See Controllers).

Illumination

There is one global lightsource in the game: the sun. This is a simple parallel light source, that casts shadows. A combination of lightmaps and shadowmaps will be used. There may also be hemispherelighting or something like that to approximate the ambient illumination of the scene.

Collision detection

Collision detection in 3D happens between the balloons and the rings. The balloons can push each other away. The rings are static obstable which block the balloons. The stucture of the objects are represented by primitive geometric objects for the collision detection. The collision with the stadium is represented by a bounding box.

Implemented Effects

- GLOW effect(1.0): For goodies and checkpoints (The next ring which should be past; the final Ring after each player has mastered the parcour)
- HDR (1.0): To make GLOW look better (Can be activated/deactivated by pressing F6)
- Shadow Maps with PCF (1.0)
- GPU-Particle System of the rockets (1.0): Can be seen by launching a rocket (see Controls)

Implementations and References

- Frustum Culling:
  - http://www.lighthouse3d.com/tutorials/view-frustum-culling/ [http://www.lighthouse3d.com/tutorials/view-frustum-culling/]

Alle Meshes werden mit einer umschließenden Box versehen, deren Ecken mit den Frustum-Planes geschnitten werden. Ist einer der Ecken innerhalb des Frustums wird das Mesh gezeichnet.

- HDR

- HDR:
  - http://cilab.knu.ac.kr/seminar/Seminar/2012/20120818%20Tone%20Mapping%20for%20HDR%20Images%20with%20Dimidiate%20Luminance%20and%20Spatial%20Distribution%20of%20Bright%20and%20Dark%20Regions.pdf [http://cilab.knu.ac.kr/seminar/Seminar/2012/20120818%20Tone%20Mapping%20for%20HDR%20Images%20with%20Dimidiate%20Luminance%20and%20Spatial%20Distribution%20of%20Bright%20and%20Dark%20Regions.pdf]
  - http://learnopengl.com/#!Advanced-Lighting/HDR [http://learnopengl.com/#!Advanced-Lighting/HDR]
  - https://lva.cg.tuwien.ac.at/cgue/wiki/lib/exe/fetch.php?media=students:cgue_hdr_ss2016.pdf [https://lva.cg.tuwien.ac.at/cgue/wiki/lib/exe/fetch.php?media=students:cgue_hdr_ss2016.pdf]
  - http://www.cmap.polytechnique.fr/~peyre/cours/x2005signal/hdr_photographic.pdf [http://www.cmap.polytechnique.fr/~peyre/cours/x2005signal/hdr_photographic.pdf]

Die Objekte werden in eine Textur gezeichnet, ohne auf den Bereich [0,1] getruncated zu werden. Im Shader wird dann eine durchschnittliche Farbe berechnet um basierd auf der Durchschnittsfarbe alle Pixel in den gültigen Bereich zu skalieren.

- Shadow Maps:
  - https://www.cg.tuwien.ac.at/courses/CG23/slides/tutorium/CG2LU_Tutorium.pdf [https://www.cg.tuwien.ac.at/courses/CG23/slides/tutorium/CG2LU_Tutorium.pdf]
  - http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping/ [http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping/]
  - https://developer.amd.com/wordpress/media/2012/10/Isidoro-ShadowMapping.pdf [https://developer.amd.com/wordpress/media/2012/10/Isidoro-ShadowMapping.pdf]

Die Shadows werden im ersten Schritt aus Sicht der Kamera in eine Depth-Textur gezeichnet. Bei eigentlichen zeichnen wir in dieser Textur nachgesehen, ob der Pixel im Schatten liegt oder nicht und jenachdem mit unterschiedlicher Helligkeit gezeichnet.

- Particle System:
  - https://www.ics.uci.edu/~gopi/CS211B/opengl_programming_guide_8th_edition.pdf [https://www.ics.uci.edu/~gopi/CS211B/opengl_programming_guide_8th_edition.pdf]
  - http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/ [http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/]
  - https://lva.cg.tuwien.ac.at/cgue/wiki/lib/exe/fetch.php?media=students:cgue_particles_ss2016.pdf [https://lva.cg.tuwien.ac.at/cgue/wiki/lib/exe/fetch.php?media=students:cgue_particles_ss2016.pdf]

Es wird ein Speicher an Partikel generiert, die bei jedem Renderzyklus weiterbewegt gezeichnet werden. Bei jedem Renderzyklus wird die Lebenszeit eines Partikels reduziert und beim Ablaufen als neuer Partikel wiederverwendet. Die Partikel werden aus Laufzeitgründen durch Instancing gezeichnet (Gleiche Objecte werden mehrmals an unterschiederschiedlichen Positionen gezeichen).

- Glow:
  - http://learnopengl.com/#!Advanced-Lighting/Bloom [http://learnopengl.com/#!Advanced-Lighting/Bloom]

Die Objekte die später glowen sollen werden in eine eigene Textur gezeichnet, dann im einem weiteren Shader mits Gauß-Filter geblurt und danach wieder mit dem Rest zusammengefügt.

- Transparency:
  - http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-10-transparency/ [http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-10-transparency/]

In das GL_BLEND Flag wird in OpenGl aktiviert und transparente Objekt mit einem Alpha-Wert < 1 gezeichnet.

Libraries

- FreeImage (Image loader): http://freeimage.sourceforge.net/ [http://freeimage.sourceforge.net/]
- Assimp (Model loader): http://www.assimp.org/ [http://www.assimp.org/]
- physix (Collisions detection): https://developer.nvidia.com/physx-sdk [https://developer.nvidia.com/physx-sdk]
- glfw (Window manegment): http://www.glfw.org/ [http://www.glfw.org/]
- glew (OpenGL Helper): http://glew.sourceforge.net/ [http://glew.sourceforge.net/]

Sketches

Left: View while playing, Middle: side view (not visible in game), Right: top view (not visible in game)