

Fish Fever – Gruppe 66

Feature Name	Description	Source (if used)
Collision Detection	We implemented collision detection using the Bullet Library. The “Player” is colliding with the other objects in the fish tank and can collect stars and make bombs explode through collision.	https://github.com/bulletphysics/bullet3 https://pybullet.org/Bullet/phpBB3/ https://github.com/kripken/bullet
Camera	We are using a camera that follows the player, using the WASD keys for movement and the mouse for looking around. Scrolling adjusts the zoom. All general settings for the camera can be set and customized.	
Adjustable parameters	Adjustments concerning cel shading, brightness, camera and the game window can be made in the settings.ini file. We recommend that you test the game in full-screen mode. Per default we activated windowed mode.	
Model classes	For storing and displaying our imported models, we implemented a mesh and model class, where the loaded models incl. textures can be stored and drawn.	https://learnopengl.com/Model-Loading/Model https://www.turbosquid.com/de/3d-models/free-golden-fish-3d-model/755156
Model Loading	We are using assimp as object loading library to read in different 3d-model files.	https://learnopengl.com/Model-Loading/Assimp
Models	We are using self-made .obj files in our project.	
Illumination model	A point light source is placed at the top of the fish tank and illuminates all objects in the scene. The material parameters can be adjusted. There is also a directional light shining from the top to further illuminate the scene. Correct normal vectors are provided.	
Image Loading	We are able to load images (e.g. for textures) using the stb_image library.	https://github.com/nothings/stb
Shaders	We are using multiple shaders: <ul style="list-style-type: none"> - general (texture.x) - collision (collision.x) 	

	<ul style="list-style-type: none"> - gui (gui.x) - procedural texture (procedural.x) - text (text.x) - bloom (multiple in bloom folder) - player + vertex anim. (fish.vert) - lightmaps (lightmap.x) - particle system (particleSystem.x) 	
Movement	The character can be moved using the WASD Keys and look around/change height (up down movement) by dragging/moving the mouse. Scrolling controls the zoom. Space bar activates boost.	
Advanced Physics	There are small pebbles dropping to the floor and rolling around. The player can interact with stars and bombs and collect them. The fish can use boost and inertia makes the movement non-linear.	
Framerate & Text Renderer	We implemented a text renderer using the freetype library that currently displays the framerate.	https://learnopengl.com/In-Practice/Text-Rendering
Framerate independency	Our game is framerate independent and is running at 60fps.	
Scene	The scene features a fishtank which acts as a boundary for the player, a fish character, multiple corals and plants, a big rock, small pebbles, a castle, a structure giving piece of wood, bombs that explode upon being hit, stars that disappear when collected.	
Win/Lose Condition	The game can be won by collecting all 5 stars and is lost when too much damage (lost all 3 lives) is taken. After winning/losing, a big white font informs the player that he/she won/lost the game. The game loop then stops after 5 seconds.	
Bloom/Glow	Bloom is applied to all the stars, to the explosions of the bombs, to the procedural coral and to the light at the top of the fish tank to give it a nice look.	https://learnopengl.com/Advanced-Lighting/Bloom
Lightmap using separate textures	Lightmaps have been created and applied to all static objects. They use the lightmap shader, where the final appearance is only dependent on the combination of the objects texture and the lightmap, with no light calculation being applied.	Sources provided in CG Tuwel course resources
CPU Particle System	We implemented a particle system that is applied to the bomb explosions and is also visible when you collect a star. The star emits star particles, and the bomb emits orange/red explosion particles.	https://youtu.be/POEzdiqEESo https://youtu.be/Rm-By2NJsrc
Procedural Texture	A procedural texture is applied to a coral on the bottom of the fish tank. It has a	https://thebookofshaders.com/11/?lan=de

	fluorescent pink look. A sine function was used to create the noise.	https://learning.oreilly.com/library/view/opengl-programming-guide/9780134495514/ch08.html https://shader.how/to/draw-a-sine-wave/ https://gist.github.com/patricio-gonza-lezvivo/670c22f3966e662d2f83
HUD	The HUD displays the health bar of the player and the collected stars. Also there is an fps display at the bottom.	
Vertex Shader Animation	We used a vertex shader animation to make the fish wiggle to make it look like the fish is swimming. When boost is activated, he wiggles faster.	
Cel Shading	Cel shading is used to give the scene a comic-like feel and make it look nice.	https://stackoverflow.com/questions/15095909/from-rgb-to-hsv-in-opengl-gsl https://www.youtube.com/watch?v=dzltGHyteng&ab_channel=ThinMatrix
Gameplay	The player controls a fish and tries to collect all the stars without colliding with the bombs. Hitting a bomb reduces health points.	

Controls

Camera movement:

- WASD for movement
- Camera movement
 - When in windowed mode: mouse dragging for rotation/navigation
 - When in fullscreen mode: mouse movement for rotation/navigation
- Scrolling wheel for zoom
- Space bar for boost

Non-Game controls:

- F1 for toggling wireframe (enable if you have trouble finding all the stars)
- F2 for toggling backface culling

- F3 for toggling visibility of collision shapes
- F4 for toggling cel shading
- F5 for toggling HUD
- F6 for toggling bloom/glow
- ESC for closing the window

How to win:

Collect all the stars. They are located on top of the big piece of wood, behind the giant rock, in the centre of the yellow coral, in one of the 3 seagrass objects in the corner next to the castle and on the castle.

Libraries

We used the following libraries in our project:

- Assimp for model loading
- stbi for image loading
- Bullet Physics
- Libraries that were previously included in the ECG framework such as GLEW, GLFW or GLM
- Freetype

We used Learnopengl as a basic resource and built our project on the ECG Framework. When the usage of ECG-classes combined with newer parts of our implementation was too cumbersome, we deleted and reimplemented given parts of the framework.

We still want to change some interactions (e.g. bomb movement) in order to provide an even better gaming experience. We also want to implement the effects lightmap using separate textures and particle system until the 24th to improve our grading. We would be very thankful for feedback concerning the currently implemented effects on if they receive the full points or if they are not sufficient, because that could make the difference between a positive and a negative grade in our case.