Magic Weasel

Implementation:

The game is written in C++. We use OpenGL 4.0. For collision detection we use the ColDet 3D library and for calculating the frame time we use GLFW. To import the Models we use the FBX SDK. Further we use GLEW for loading OpenGL Extensions and for playing sound we use the sound library Irrklang.
To control the weasel we use an Xbox 360–Controller. To use the Xbox Controller we use XInput.

Gameplay:

The game is a comic-style jump'n'run / platformer. You play a weasel with a baseball bat. The goal is to collect all the crystals scattered around the level. The glow effect of the crystals helps you to find them. Some of the aliens are also carrying crystals with them. They won't give it to you willingly though, so you have to whack them with your baseball bat to get to it. That means avoiding aliens is not an option to win the game. Further you have to jump from platform to platform to get the remaining crystals. We also have implemented moving platforms.

You start with three hearts. If you lose them, you die and have to start all over again. But there are several items floating around to help you.

- Hearts will replenish your life, if you have been hit.
- Touch the floating party hats in order to gain the duck power! There's nothing aliens hate more than a vicious rubber duck! For a short time period you will be invincible and can just run over the aliens, but be aware: as soon as the effect wears off, you're vulnerable again!

AI:

For the enemies we used a simple AI. Each alien has 3 Waypoints. It walks between its 3 waypoints until the player reaches a determined distance to the aliens. Then the player is followed by them. The aliens only move, if you get close. They are a nervous bunch and get very excited at the hero's approach. This was necessary to have enough performance in our game, because when every alien would walk all the time, then the frame rate would drop.

Camera:

We use a third-person camera, but you can freely move the camera around with the right control stick.

Controls:

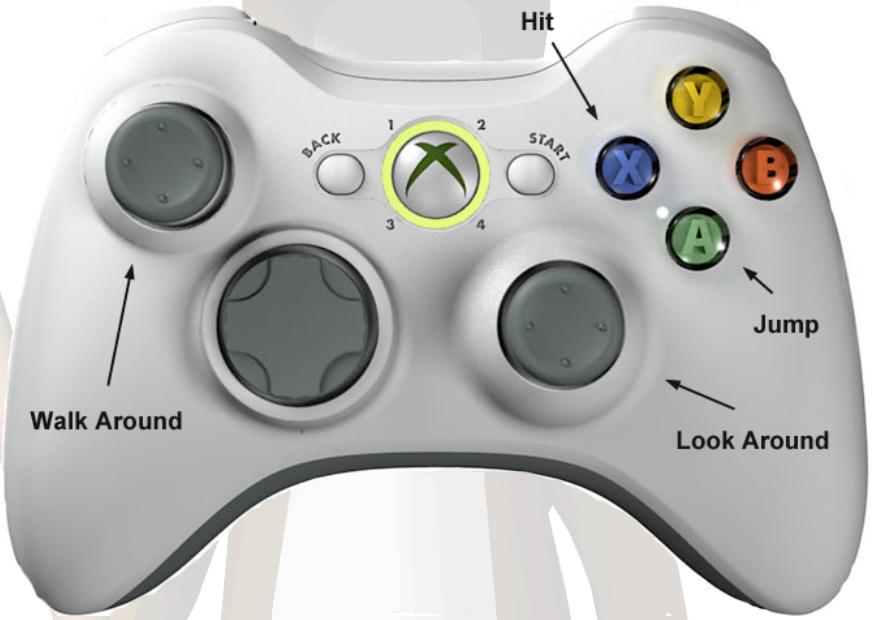
ATTENTION! : You can play the game with the keyboard, but it is a lot better to play it with a controller, because it is not made for the keyboard!

Walk Around: You can walk around with the left analog stick or with the WASD keys on the keyboard.

Look Around: You can look around with the right analog stick or with the mouse.

Jump: You can jump with the A-button or with the space bar on the keyboard.

Hit: You can hit with the X-button or the left mouse button.



Features:

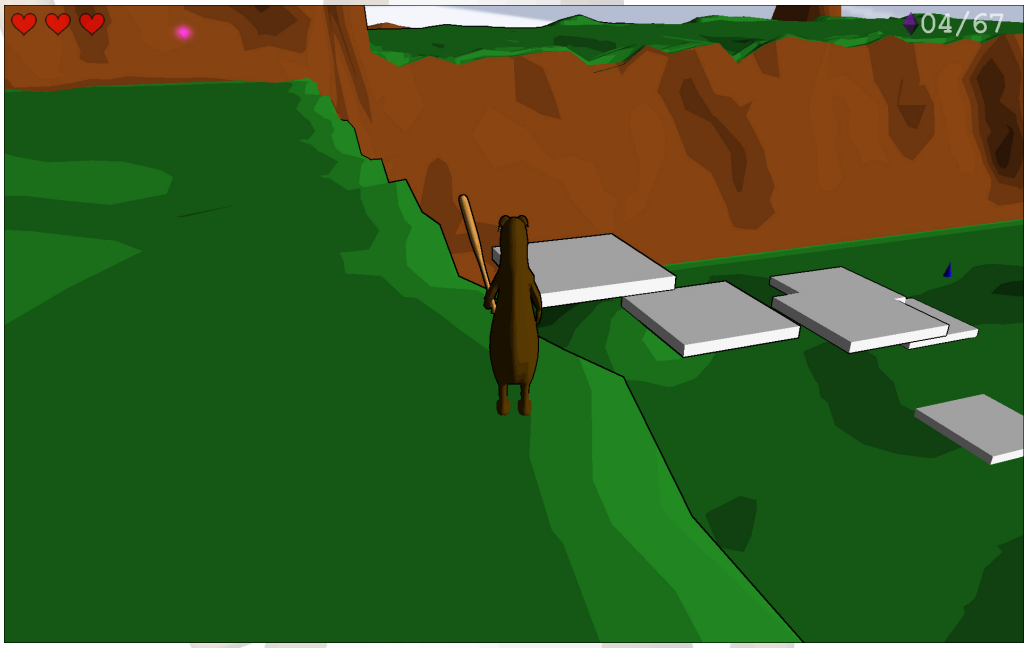
All models are self-made, even the landscape. They are modeled in Autodesk Maya. For setting the positions of the aliens and the crystals we load the scene into 3DS Max and wrote a script in Maxscript to export the positions in a text file, which we load in our engine. Also all of the textures are self-made. For this we used Paint.net and Photoshop. We also have used a high-level game sound engine called irrKlang so you can enjoy some music and simple 2D-sound effects, which make the gaming experience more lively. The following tools were used to create sound:

- Guitar Pro 5 to create music (except the duck mode song, which is borrowed from Squaresoft's Final Fantasy 8)
- Audacity to re-master sound effects / music

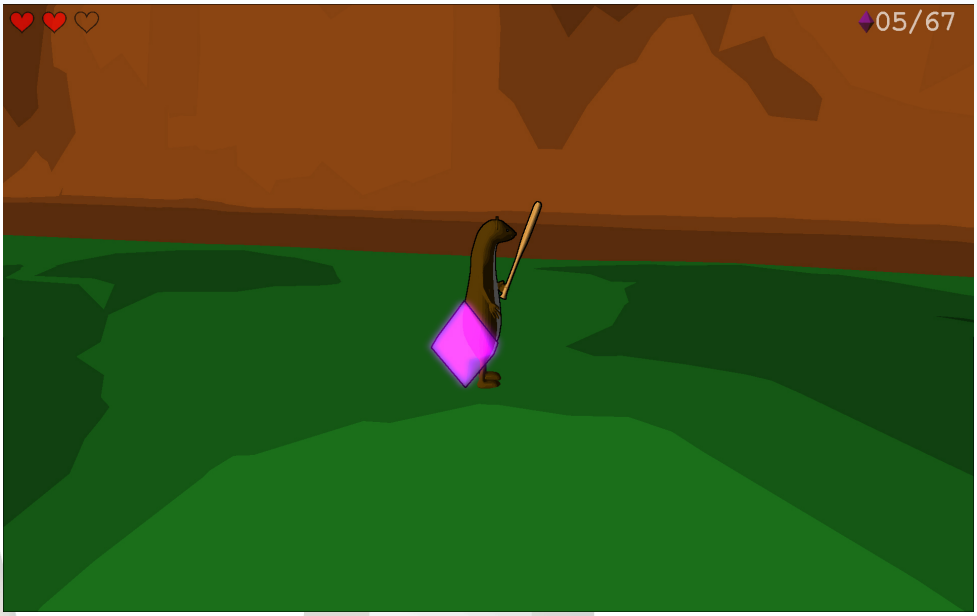
The background sound of the game is self-made.
We have got our own, simple HUD, which shows your current hearts and collected crystals.

Effects:

We implemented a toon shader, which is used for all models except for the skybox (we made a toon-like texture for it in Photoshop). For this we used edge detection. The objects are first rendered with discrete light levels into a frame buffer object. We render the color image and the depth image at one render pass and in a second render pass, we load the depth texture in the shader and use an edge detection filter to make black contours along the edges. Then we render only the edges into a texture and in the next render pass, both the original color texture and the texture with the edges are rendered on the screen with additive blending. We have implemented this effect according to the tutorials listed in the chapter “Sources”, but for the edge detection we used the depth texture instead of the color texture and a Sobel operator. The following figure shows a screenshot, which shows the effect:



Further we implemented glow. It is used for the crystals to make it easier to find them. For this we render to a third texture in the first render pass, but only with the crystals in it. The other objects are coloured black. Then in a second pass we render that texture into a smaller texture. In the next pass we blur the texture in the horizontal direction and in the next render pass we blur the texture vertical. Then in the final render pass, we render this texture also on the screen using alpha blending. For the glow effect we used the lecture slides.
The crystals are also rendered transparent. You can see the glow effect in the following screenshot.



The third effect we implemented is GPU Vertex Skinning. We use this effect for the enemies (aliens) and for our weasel. We made the animation in Maya and then imported the animation into our engine. With the FBX-SDK we got the skinning matrix, the bones and the skinning weights, which we load into the vertex shader. We have implemented this effect according to the tutorial listed in the chapter "Sources".

We also implemented View Frustum Culling. You can see the effect in the debug window, which shows you how many objects are culled and which effect it has on the frame rate.

Light and Textures:

All of the models are lightened and textured, except for the sky, which is only textured. We have implemented ambient and diffuse light. We have one directional light source, the sun. All of the textures are self-made.

Achievements:

We included the duck and the party hat, self-made textures and a self-made background sound. When the player collects a party hat the weasel is transformed into a duck.

OpenGL Features:

Upon start a console window is opened, which shows the effect of the culling, the frame rate and the controls for switching on/off the OpenGL features. In the following table you can see the controls for this:

Key	Effect
F2	Frame time on/off
F3	Wireframe on/off
F4	Textur-Sampling-Quality: Nearest Neighbor/Bilinear

F5	Mip Mapping-Quality: Off/Nearest Neighbor/Linear
F8	View frustum culling on/off
F9	Transparency on/off

Configuration:

We have made a configuration file named config.txt in the PositionFiles folder. When you write the number "1" in it, the game is rendered in full screen mode and if you write the number "0" in it, the game is rendered in a smaller window.

Modelling Tools & Libraries

Tools used for model creation

- Autodesk Maya for model creation /-animation (Models are FBX-files).
- Paint .NET/GIMP to export TARGA-Files for textures.
- Photoshop

We have used the following libraries in our Project:

- GLEW (1.9.0, win32)
- GLFW (2.7.7, win32)
- coldet (library for simple collision detection)
- irrklang (high-level game-sound api)
- FBX SDK

Sources

Tutorials

FBX SDK Documentation: http://docs.autodesk.com/FBX/2013/ENU/FBX-SDK-Documentation/index.html
HUD: http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-11-2d-text/
Toon Shader: http://www.lighthouse3d.com/tutorials/glsl-tutorial/toon-shading/
Edge Detection: http://coding-experiments.blogspot.co.at/2010/06/edge-detection.html
Vertex Skinning: http://silk-ge.blogspot.co.at/2010/11/tutorial-vertex-skinning.html
Glow: Repetetorium Slides
http://www.opengl-tutorial.org/
http://www.rastertek.com/

Tools

Autodesk Maya 2013 - http://www.autodesk.de/products/autodesk-maya/overview
Paint .NET - http://www.getpaint.net/
GIMP - http://www.gimp.org/
Guitar Pro - www.guitar-pro.com/

Audacity - http://audacity.sourceforge.net/
Photoshop - http://www.photoshop.com/
3ds Max - http://www.autodesk.com/products/autodesk-3ds-max/overview

Libraries
GLEW - http://glew.sourceforge.net/
GLFW - http://www.glfw.org/
irrklang - http://www.ambiera.com/irrklang/
coldet - http://coldetdotnet.sourceforge.net/
FBX SDK - http://docs.autodesk.com/FBX/2013/ENU/FBX-SDK-Documentation/index.html

Sound
Sound effects are all from http://www.freesound.org/.
The duck mode-song is Odeka de Chocobo from the Final Fantasy 8 Original Soundtrack.