# 2nd Submission

## new features (since 1st submission)

**collision detection**
Man kann nicht mehr (wie bei der letzten Abgabe) durch Wände oder Objekte gehen.

**ui**
Wir haben eine UI (oben rechts) implementiert, die anzeigt wie weit Jasper von dem Verfolger (der Katze) entfernt ist. Die UI ändert sich aufgrund der Distanz.

**sound**
Seit dem letzten mal hat unser Spiel auch Sound! Sounds für Springen, schneller werden, Schleime aufsammeln, für die Katze und das Menü existieren. Der Sound im Hintergrund wird schneller und lauter, sobald die Distanz zum Verfolger kleiner wird.

**slime pickup**
Die grünen Schleime am Boden machen Jasper beim aufheben schneller und respawnen nach 5 Sekunden wieder.

**gameover screens**
Wenn man eingeholt wird, in ein Loch fällt oder das Level beendet, wird ein gameover-screen angezeigt.

**cat**
Der Verfolger ist eine Katze, die “in der Vorstellung von Jasper” irgendein böses Monster ist. Deswegen wird die Katze bei größerer Entfernung als “böser” von unserem scheuen Geist wahrgenommen.

**godmode**
Wenn man am Anfang den godmode auswählt, kann man fliegen, durch das drücken der linken Shift-Taste unendlich lang schneller sein und das Spiel wird nicht beendet wenn man eingeholt wird bzw. durch ein Loch fällt. (Eignet sich eventuell für Tutoren. ;) )

## effects

**light mapping - 0.5P**
Die Lightmaps werden von unserem Modellierungsprogramm (Cinema4D) gebacken und anschließend im Projekt wie eine normale Textur auf das Level gemappt.
Bevor das Spiel startet, kann man auswählen ob man High-Res oder Low-Res Texturen verwenden will. (Für Laptops sind eher Low-Res vorgeschlagen.)

**motion blur - 1.5P**
Wir haben uns an den Tutorials von Chapman und GPU Gems für screen motionblur orientiert. Insgesamt ist es so implementiert, dass für normales Movement ein leichter Motionblur und für schnelleres Movement ein Motionblur mit längeren "tails" angezeigt wird. Der Motionblur ist unabhängig von der Framerate.

**spotlights - 0.5P**
Ein Spotlight (Eine Unterklasse von Scene/Lights/Light) ist ein Pointlight mit zusätzlichen Parametern: outer cone, inner cone und direction. Diese werden den Shadern der dynamischen Objekte (hauptsächlich zu sehen bei Jasper und der Katze) übergeben und im jeweiligen Fragmentshader (z.B. toon.frag) werden sie dann mit einberechnet.

**toon shader - 0.5P**
Nach dem normalen shading jedes Fragments wird der diffuse und der spekuläre Farbteil im jeweiligen Fragment shader aufgeteilt, damit der charakteristische Toon Effekt zustande kommt.

**animated textures - 1P**
In AnimatedTexture.cpp wird ein .avi File geladen. Bei jedem Bind der Textur wird das aktuelle Frame abgefragt und abhängig davon dann das passende Frame aus dem Video genommen und als Textur dargestellt. Die Animated Texture wird bei der schwarzen Textur Katze angewandt, bei der der Effekt von Blut rinnen simuliert werden soll (auch wenn das jetzt nicht unbedingt so ausschaut, weil die Animation einfach schlecht gemacht ist ^^).

# implementation

**complex objects**
Wie unschwer zu erkennen ist, gibt es in unserem Spiel komplexere Objekte, die mit Cinema 4D erstellt oder aus dem Internet geladen und dann mit Assimp importiert wurden.

**animated object**
Es wurde eine einfache Animation für das Drehen der Räder des Skateboards, auf dem die Katze fährt erzeugt. Diese wurde mittels Assimp geladen und durch Interpolation zwischen den Keyframes auf die dazugehörigen Meshes angewandt (in AnimatedObject.cpp). In unserem Fall besteht das Objekt aus dem Mesh Brett und den 2 "Räder" Meshes , wobei die Animation nur auf den Rädern durchgeführt wird.

**view-frustum-culling**
Es werden nur Objekte in Renderer.cpp gerendert, wenn die Boundingboxes von mindestens ⅓ der Meshes eines Objekts im Frustum liegen. Jasper und das Level werden immer gerendert.

**transparency**
Transparenz wird in unserem Spiel von 2 verschiedenen Objekten verwendet: von Jasper (da er ein Geist ist) und von den Slimes (da ektoplasmischer Schleim natürlich durchsichtig ist, wie sollte er sonst aussehen?!?)

**experimenting with opengl**

- Vertex-Buffer-Objects (VBO) - Werden für positions, etc. verwendet.
- Vertex Array Objects (VAO) - Werden für jedes gezeichnete Objekt verwendet.
- FBO (Frame Buffer Object) - Verwendet in "Motionblur.cpp" für Postprocessing Motionblur
- Mip Mapping (on/off), Textur-Sampling-Quality (Bi/Trilinear Filtering) und Wireframe Mode - können wie im Menu beschrieben geändert werden. (Allerdings wird die Videotextur durch das verändern der Textur-Sampling-Quality unkenntlich)

## libraries and programs

- assimp (Um .obj-Files zu laden)
- FreeImage (Um Bilder zu laden)
- glm (Geometrie)
- glew32 (OpenGl Extension)
- glfw3 (Window Manager)
- Bullet (Physics)
- irrKlang (Sound)
- Cinema 4D (for modeling)
- Photoshop CS6 (for UI and other screens)

## websites

- https://www.freesound.org/ - Sounds
- http://tf3dm.com/ - 3D Models
- http://john-chapman-graphics.blogspot.co.at/2013/01/what-is-motion-blur-motion-pictures-are.html - motion blur
- http://http.developer.nvidia.com/GPUGems3/gpugems3_ch27.html - motion blur
- http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-11-2d-text/ - 2D Text
- http://www.lighthouse3d.com/tutorials/view-frustum-culling/ - View Frustum Culling
- http://nehe.gamedev.net/tutorial/playing_avi_files_in_opengl/23001/ - Animated Texture
- http://gamedev.stackexchange.com/questions/26382/i-cant-figure-out-how-to-animate-my-loaded-model-with-assimp - Animated Object

**VisLab testing:** Cylon - AMD Radeon: [X], Cylon - NVidia GeForce: [✓]