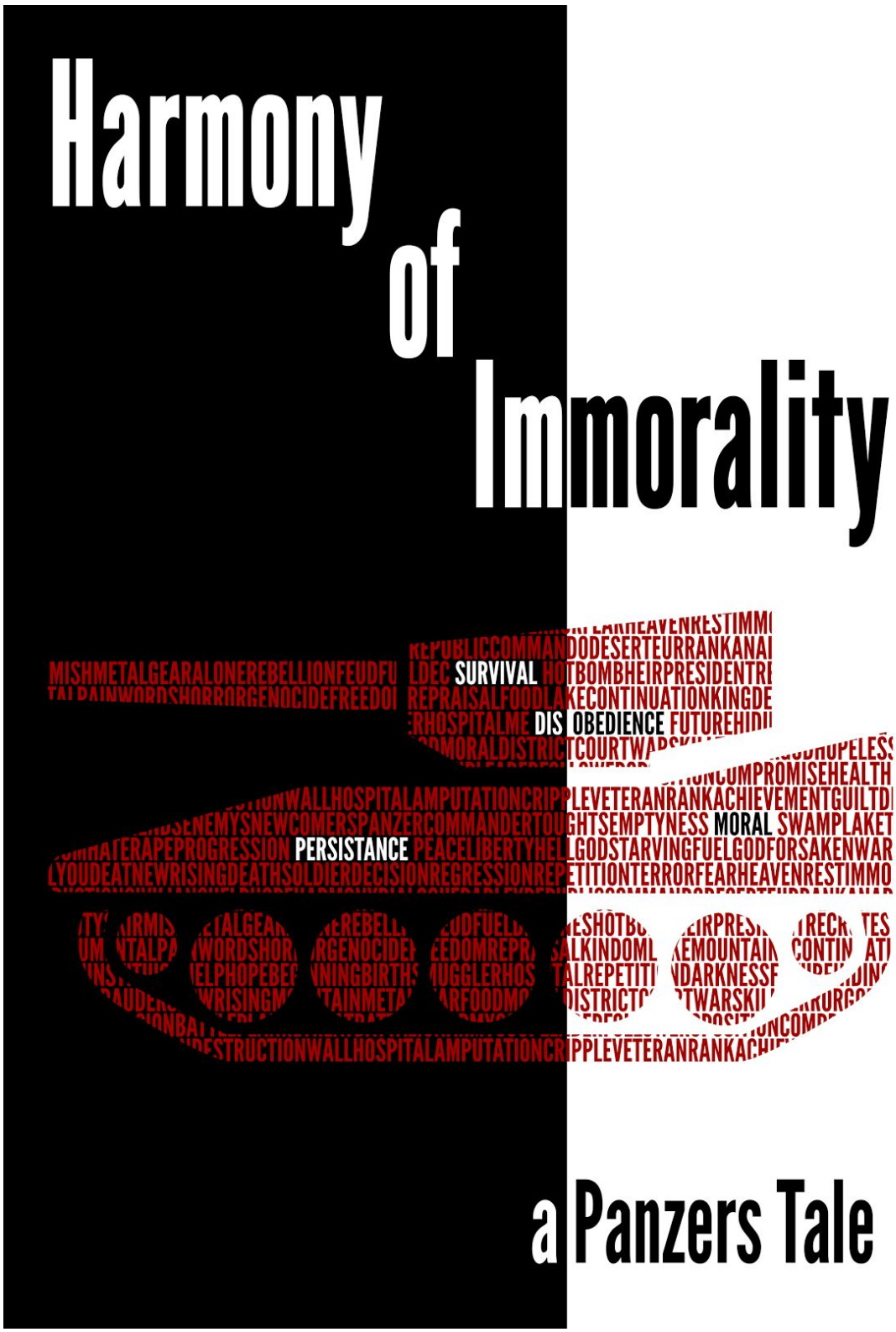
186.831 UE Computergraphik
Harmony of Immorality
Gabriel Rauter 1026292 033 532
Andreas Seiwaldstätter 1025541 033 534



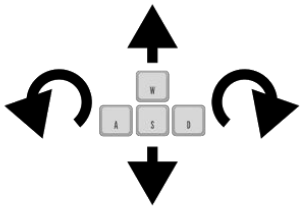
Dokumentation 2. Abgabe

186.831 UE Computergraphik
Harmony of Immorality
Gabriel Rauter 1026292 033 532
Andreas Seiwaldstätter 1025541 033 534

Quickstart

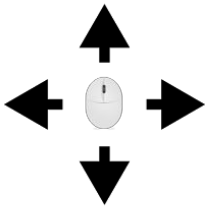
Movement

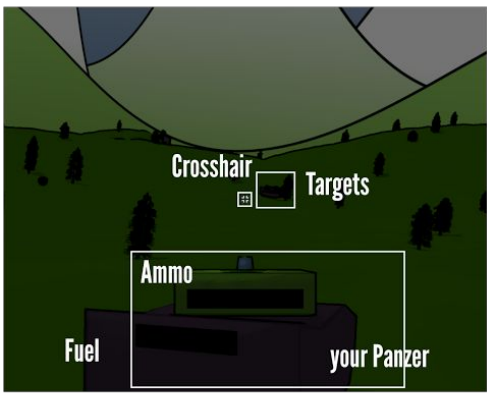
costs fuel
no fuel ->
GAME OVER!



Aiming

with mouse
shooting with a click
shooting costs ammo





Move arround, shoot Targets.
Mooving costs Fuel. Time costs Moral.
As Moral decreases the World gets darker.
If it is to dark to see. You lose.

Destroy 3 Targets to win a level.

Gabriel Rauter 1026292 033 532
Andreas Seiwaldstätter 1025541 033 534

Features von Abgabe 1 (aktualisiert)

1. Free movable camera

Die Kamera im Spiel schaut immer in dieselbe Richtung wie die Kanone des Panzers. Diese dreht sich mit dem Turm mit. Der Turm kann über 360° gedreht werden. Nach oben und unten ist die Drehung begrenzt. Gedreht wird, wie üblich mit Bewegen der Maus.
Außerdem gibt es eine frei bewegliche Kamera, auf die mittels der Taste “V” gewechselt werden kann. Mit dieser ist es möglich sich schneller durch die Landschaft zu bewegen.

2. Moving objects

Die Spielfigur wird vom Spieler durch die Tasten WASD bewegt, andere Panzer bewegen sich von selbst zufällig durch die Welt.

Steine rollen von der Physik bedingt durch die Landschaft. Diese können auch angestoßen werden.

3. Texture Mapping

Die Panzer, sowie das Fadenkreuz, nutzen Texture Mapping über UV Koordinaten. Die UV Koordinaten werden über den Model Loader aus collada Files geladen. Die Texturen über libpng. Es ist möglich mit F4 zwischen mehreren Textur-Sampling-Qualities durchzuschalten: Nearest Neighbor und Bilinear.

4. Simple lighting and materials

Als Lichtquelle wird nur eine Sonne verwendet. Dies entspricht einem gerichtetem Licht. Materialien werden aus der Model-Datei geladen und der Datenstruktur der Engine hinzugefügt sowie im einfachen Cell-Shader verwendet. Die Normalen werden auch über die Model Files geladen.
Speculars kann man an Steinen und dem Panzer sehen, alle Objekte werfen Schatten, Transparenz ist beim Partikeleffekt sichtbar.

5. Controls

Die Spielfigur, der Panzer, ist mit W A S D steuerbar. W und S werden für Vor- und Rückwärtsbewegung des Panzers genutzt. A und D drehen den ganzen Panzer gegen bzw. im Uhrzeigersinn. Mit der horizontalen Achse der Maus lässt sich der Turm des Panzers entlang der Y-Achse drehen und mit der vertikalen Achse der Maus lässt sich die Kanone des Panzers nach oben bzw. nach unten schwenken. Mit einem linken Mausklick lassen sich gegnerische Panzer "abschießen". Mit der Pfeiltasten kann die Position der Sonne geändert werden.

6. Basic Gameplay

Der Panzer ist mit WASD steuerbar, mit Linksklick kann geschossen werden. Dabei wird von Bullet ein Ray in Richtung der Kanone gecastet, der einen Hitpoint zurückliefert (falls so einer existiert). Das Objekt, das in unmittelbarer Nähe des Hitpoints ist, wird zerstört

(aus der Liste der zu zeichnenden Objekte und aus der Physik gelöscht). Außerdem erscheint ein Partikeleffekt an der getroffenen Stelle.
Es gibt bereits eine Lose-Condition. Verloren hat man wenn die Moral oder der Tank des Panzers auf 0 sinkt (die Moral wird durch die Helligkeit der Landschaft angegeben; die Tankanzeige befindet sich am Panzer). Auch die Munition wird aktualisiert (Anzeige am Turret). Bei 0 Schuss kann nichts mehr zerstört werden. Die Moral sinkt mit der Zeit, der Tank bei Bewegung und Rotation. Außerdem wurden Sounds bei Bewegung, Schuss und Treffer wiedergegeben.
Der Panzer verhält sich nun korrekt wie ein Vehicle. Dazu wurde die Bullet Physik verfeinert
Die Score steigt beim Abschuss von Zielen (Panzern). Es gibt keine Win-Condition mehr. Man spielt bis man verloren hat und versucht einen möglichst oft ins nächste Level aufzusteigen.

Gabriel Rauter 1026292 033 532
Andreas Seiwaldstätter 1025541 033 534

Features Abgabe 2

1. Effects

- Cell Shading
  an allen Objekten sichtbar.
- + Contours (edge detection)
  per post-processing mit sobel und gaussfilter
- Shadow Maps
  + PCF aber leider keine Shadow-Frustum
- GPU-Particle System (+Transform Feedback, Instancing)
  sichtbar als Explosion durch Kollision des Schusses mit einem Kollisionsobjekt.

2. Complex Objects

Alle Objekte wurden selbst in Blender als Model umgesetzt. Es gibt Gebäude, Steine, sowie Bäume.

3. Animated Objects

Der Turm des Panzers dreht sich getrennt von der Karosserie.

4. View-Frustum-Culling

View-Frustum-Culling wurde implementiert und ist mit F8 ein-/ausschaltbar. Die Reduzierung der drawCalls kann man sich mit F7 ansehen.

5. Transparency

Transparenz ist beim Crosshair (volle Transparenz) sowie bei den Partikeln (Semi Transparenz) zu finden.

6. Experimenting with OpenGL

Folgende OpenGL Features wurden verwendet:

- Vertex-Buffer-Objects (VBO)
- Vertex Array Objects (VAO)
- Buffer-Objects: FBO (Frame Buffer Object) or UBO (Uniform Buffer Object)
- Mip Mapping (on/off)
- Textur-Sampling-Quality (Bi/Trilinear Filtering)

Die gewünschten Funktionstasten wurden implementiert und erweitert:
F1 - Help (if available)
F2 - Frame Time on/off
F3 - Wire Frame on/off
F4 - Textur-Sampling-Quality: Nearest Neighbor/Bilinear
F5 - Mip Mapping-Quality: Off/Nearest Neighbor/Linear
F6 - Shadow on/off

F7 - Draw Calls on/off
F8 - Viewfrustum-Culling on/off
F9 - Transparency on/off
F10 - PostProcessing on/off

Weitere “Features”:

Die Bullet Library wird für Kollisionserkennung verwendet.
Das Spiel läuft auch auf Mac Os X 10.10.3 und Arch Linux und lässt sich dort über cmake und clang kompilieren.

Verwendete Software

Visual Studio 2013, Gnome Builder, Atom Editor, Notepad++, Vim, Blender, Paint, Inkscape, Gimp.

Used Librarys

Bullet Library Collision-Detection, Movement & Physic (http://bulletphysics.org/)
libpng for texture loading (www.libpng.org)
png++ for texture loading (http://savannah.nongnu.org/projects/pngpp/)
assimp for model loading (http://assimp.sourceforge.net/)
glfw for opengl context (http://www.glfw.org/)
glew for opengl context (http://glew.sourceforge.net/)
glm used for openGL Math (http://glm.g-truc.net/)
zlib used by libpng (http://www.zlib.net/)
tclap a command line parser library (http://tclap.sourceforge.net/)
sfml audio module for 3d sound and music (http://www.sfml-dev.org/)
sfml abhängigkeiten: openal, ogg, vorbis, flac, winmm(nur auf Windows)