
Dokumentation

Mr.Puff in Danger



Entwickler

Name	Matrikelnummer	E-Mail
Stephan Sgarz	11910593	e11910593@student.tuwien.ac.at
Kevin Baur	11827180	e11827180@student.tuwien.ac.at

Steuerung

Taste	Effekt
W, A, S, D	Mr.Puff kontrollieren
Maus linksklick	Kamera frei verändern
Maus rechtsklick	Zurücksetzen der Kamera hinter den Spieler
M	Musik an/aus
Mausrad vorne/hinten	Zoomen zu Mr.Puff oder weg
F	Licht einschalten
ESC	Spiel beenden
C	Steuerung anzeigen

R	Neustart nachdem Mr.Puff runtergefallen ist
---	---

Weitere Tastenbelegung

Taste	Effekt
F1	Batterie auf 300 Sekunden setzen (zum debuggen)
N	Normal Mapping an/aus
H	Hud ein/aus
G	Credits anzeigen/ausblenden
I	FPS zähler ein/aus

Implementation

Unsere Implementierung basiert auf das von der Lehrveranstaltung bereitgestellte Einführung in die Computergrafik (ECG) Framework. Wir haben uns dazu entschieden, da einige Teile der Viewing Pipeline bereits vorhanden waren und wir so schnell in unser Projekt finden konnten.

Die Kamera wurde von uns neu implementiert. Es wurde eine 3D-Kamera implementiert wie man sie aus Rollenspielen kennt. Es ist als eine Kamera in dritter Person Sicht. Dabei wird die Kamera an die Position des Spielers fixiert und folgt diesem in konstantem Abstand. Mit dem Mausrad kann der Abstand verändert werden. Bewegt sich das Spielobjekt nach links oder rechts, wird der Winkel zwischen Spieler und Kamera berechnet und dementsprechend berichtigt. So bleibt die Kamera immer im selben Winkel hinter dem Spieler. Mit der linken Maustaste kann der Pitch verändert werden, so kann die "Höhe" der Kamera verändert werden. Mit der rechten Maustaste kann der Yaw(x-Achse) verändert werden, damit soll eine Rundumsicht ermöglicht werden, dann aber bei dem loslassen der Taste wieder zurückkehren, um die Spielbarkeit zu erleichtern.

Aufgrund der vielen Tutorials, die wir online zu Bullet gefunden haben, entschieden wir uns für diese Physics Engine. Zuerst erstellen wir eine Physics-Welt mit Schwerkraft und fügen anschließend verschiedene Körper, die entweder statisch (Masse 0) oder rigid sind, in die Welt ein.

In jedem Renderloop zeichnen wir nun passende Meshes and die jeweilige Position des dazugehörigen Bullet-Objects. Diese Meshes werden zusätzlich mit Texturen versehen und von 4 verschiedenen Lichtquellen, 3 Point-Lights und einem Directional-Light, beleuchtet. Hierbei werden verschiedene Shader verwendet, da die Beleuchtung zwischen Objekten in unserem Spiel stark variiert. Zusätzlich setzten wir zu jedem Objekt spezifische Materialkoeffizienten, die die Beleuchtung individuell verändert.

Wir haben die alle komplexeren Modelle selber in Blender modelliert, außer Schaufel und Rechen, und mit verschiedenen Texturen zusammen-gebaked. Diese werden mithilfe von assimp in das Spiel geladen und an Bullet-Objekte angebunden.

Zusätzlich rendern wir auch eine Skybox, die eine Strandszene darstellt und die Immersion in unser Spiel verstärkt.

Mithilfe der freetype Bibliothek wird Text gerendert. So können dem/der Spieler/in wichtige Informationen im Spiel angezeigt werden, wie zum Beispiel die verbleibende Zeit des Lichts in Form einer Glühbirne, oder die Win/Lose Kondition. Außerdem werden dem Spieler die verbleibenden Herzen als Leben angezeigt. Mr.Puff kann insgesamt 4 Mal in den Abgrund stürzen, bevor das Spiel vorbei ist. Mit der Taste C kann eine schnelle Übersicht der Steuerung angezeigt werden.

Die Steuerung unseres Spiels funktioniert mittels Key-Bindings, auf die während des Renderloops zugegriffen werden kann. So können wir mithilfe von "F" beispielsweise das Licht aus- und einschalten. Wird "F" gedrückt, so schalten die Shader auf eine dunklere beziehungsweise gar keine Lichtwahrnehmung um. Mithilfe von "W" und "S" kann der Player-Character nach vorne und hinten bewegt werden. Mit "A" und "D" wird der Spieler samt Kamera nach links oder rechts gedreht. Verliert der/die Spieler/in ein Herz, kann mit "R" ein neuer Versuch gestartet werden.

Das Ambiente wird durch einen spannenden Soundtrack abgerundet. Dazu wird die Bibliothek irrKlang verwendet. Es kann im Main Menü zwischen einer Epic-Quest-Musik <https://opengameart.org/content/soliloquy> oder einem Metal Song gewählt werden <https://opengameart.org/content/heavy-metal-riffs-3hrmt3>.

Wie bereits erwähnt gibt es auch ein Main Menu, in dem man zwischen Fullscreen oder nicht und eben der Soundtrack gewählt werden kann. Anschließend startet dann das Spiel mit den gewünschten Einstellungen.

Zusätzliche Bibliotheken

Folgende zusätzliche Bibliotheken wurden in das ECG Framework eingebunden:

- assimp (zum Laden von .obj Modellen) <https://www.assimp.org/>
- bulletCollision, bulletDynamics, LinearMath (Physics) <https://pybullet.org/wordpress/>
- freetype (generierungen von Text) <https://www.freetype.org/index.html>
- irrKlang (sound) <https://www.ambiera.com/irrklang/>

Features

Gameplay:

3D Geometry	Wir laden mit Assimp mehrere Objekte in unsere Spielszene. Darunter die Spielfigur Mr.Puff, das Schloss, der Weg über den Mr.Puff zum Ende gelangen soll, alle aufnehmbaren Objekte (Herzen, Batterie), Alle Fenster und das Tor am Ende sowie auch die Säulen. Alle genannten Objekte wurden von uns selbst in Blender erstellt und mit Texturen versehen (diffuse, normal, lightmap).
Playable	Unser Spiel ist spielbar. Mit Hilfe der Maus kann die 3D-Kamera bedient werden. Mit der Tastatur kann das Spielobjekt gesteuert und andere nützliche Fähigkeiten eingesetzt werden (Lampe, Neustart,..).
Advanced Gameplay	Es gibt Cubes, die den Weg versperren und verschoben werden müssen um auf Plattformen zu gelangen. Man kann Items mithilfe von Collision Detection einsammeln die einen Nutzen bringen (Extra Leben oder Batterie für Licht).
Min. 60 FPS / Framrate independent	Unser Spiel liefert mindestens 60 Frames per Seconds und ist Frame-Unabhängig. Die Bewegungen von Mr.Puff (gehen, drehen, springen) und auch die Gravitation der Szene werden mit der deltaTime multipliziert (zu finden in Camera.cpp methode updatePlayerPosition(), PhysicsObjet.cpp methode renderSphere() und durch die physics world indem wir die stepSimulation mit der deltaTime aufrufen).
Win/Lose Condition	Erreicht man die andere Seite des Labyrinths hat man gewonnen, fällt man 4 Mal runter ist das Spiel verloren.
Intuitive Controls	Steuerung vom Spielobjekt mit WASD, springen mit Leertaste, Kamera mit der Maus.
Intuitive Camera	3D-Kamera, mit der linken Maustaste kann frei verändert werden. Mit der rechten Maustaste wird die Kamera direkt hinter den Spieler zurückverschoben. Das soll dem Spieler helfen die Umgebung besser zu erkunden und schnell wieder in der Ausgangsposition mit der richtigen Bewegungsrichtung zu sein. Mit dem Mausrad kann hin oder zurück gescrollt werden.
Illumination	Wir haben in unserem Spiel 4 Lichtquellen. 3 Pointlights und ein ambientes Licht. Jedes Objekt hat mindestens ein zugewiesenes Material. Jedes Objekt wird mit

	Normalvektoren berechnet.
Textures	Mr. Puff wird eine Texture zugewiesen. Die Textur ist mit mipmapping und triangular filtering implementiert.
Moving Objects	Im Spiel gibt es natürlich die Spielfigur Mr.Puff die bewegt werden kann, aber auch 3 Würfel die mit Procedural Textures versehen sind, sowie auch 1 Würfel mit normal Texturen, der verschoben werden kann um ein Extra Leben einzusammeln.
Dokumentation	Sehen sie hier.
Fullscreen capability	Bevor man das Spiel als Fullscreen starten möchte, müssen im <i>settingsFullscreen.ini</i> file die richtigen Bildschirm Maße angegeben werden. Hier kann auch die Auflösung angepasst werden.
Optional	
Collision Detection	Wir verwenden die physics Engine Bullet. Der Spielcharakter kann springen sobald er mit dem Untergrund kollidiert und kann sich durch keine Physics-Objekte durchbewegen.
Advanced Physics	Es gibt Würfel, die verschoben werden können und auch runtergeschmissen werden können. Außerdem können auch Items eingesammelt werden, die dem Spieler/der Spielerin helfen können.
Heads-Up Display	Display mit Anzeige wie viele Leben noch verfügbar sind, ob das Licht eingeschaltet ist, wieviel Batterie noch übrig ist und generelle Informationen wie Steuerung oder das Ziel des Spiels. Kann mit H ausgeschaltet oder eingeschaltet werden.

Effekte:

Lightmap using Seperate Textures	Alle unsere statischen Objekte wurden mit Blender extern gebaked und deren Lightmap in unseren Shadern mit der diffusen Textur zur Laufzeit gemischt. Kann zum Beispiel im <i>simpleLightmappingShader.frag</i> nachgeschaut werden. Die Lightmaps befinden sich im assets Ordner mit den zugehörigen Objekten.
CPU Particle System	In Klasse <i>ParticleSystem.cpp</i> zu finden. Das Particle System erschafft pro Frame Sandkörner mittels instancing, die von der Decke rieseln. Jedes Korn hat hierbei seine eigenen Werte und Positionen, die von der CPU in jedem Renderloop upgedated werden.
Procedural Texture	In Klasse <i>ProceduralTexture.cpp</i> zu finden. Die Textur der Sandsteine, die Mr.Puff den Weg versperren, wird in dem Shader mittels der Simplex Noise Funktion berechnet.
Simple Normal Mapping	Im Shader File <i>normalMapping2.frag</i> . Ein- und Ausschaltbar mit der Taste N. Der Felsen bekommt Eine Diffuse und eine Normale Textur mithilfe dessen wir uns die einzelnen Lichtkomponenten berechnen.
Bloom/Glow	Zu finden in der Klasse <i>Bloom.cpp</i> . Hier rendern wir einmal die besonders hellen Lichtstellen auf einen Quad und verwenden einen Blur Filter um den Bloom-Effekt zu realisieren. Diesen legen wir anschließen über den normalen Output.

Referenzen mit denen die Effekte und Gameplay Features implementiert wurden:

Billboards:

<http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/billboards/>

Particles:

<http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/>

Lightmaps:

<http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-15-lightmaps/>

Normal Mapping:

<http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-13-normal-mapping/>

<https://ogldev.org/www/tutorial26/tutorial26.html>

Blender

<https://www.youtube.com/watch?v=M0YQPtkOi8Y&t=208s>

Text Rendering

<https://learnopengl.com/In-Practice/Text-Rendering>

Bloom/Glow

<https://sudonull.com/post/12426-Learn-OpenGL-Lesson-58-Bloom>

Bullet

<https://pybullet.org/wordpress/index.php/forum-2/>

https://github.com/222464/EvolvedVirtualCreaturesRepo/blob/master/VirtualCreatures/Volumetric_SDL/Source/SceneObjects/Physics/DynamicCharacterController.cpp

<https://www.youtube.com/watch?v=YweNArzAHs4&t=1085s>

Texturen

<https://3dtextures.me/>

Musik

<https://opengameart.org/content/heavy-metal-riffs-3hrmt3>

<https://opengameart.org/content/soliloquy>

Camera

https://www.youtube.com/watch?v=d-kuzyCkjoQ&list=RDCMUCUkRj4qoT1bsWpE_C8IZYoQ&index=2

<https://www.youtube.com/watch?v=PoxDDZmctnU&t=1s>

3D Modelle

<https://free3d.com/de/3d-model/shovel-v1--914677.html>

<https://free3d.com/de/3d-model/toy-hand-rake-v1--503126.html>