Submission 2 Documentation

Gameplay

Playable
Player (Kinematic Objekt) wird automatisch mithilfe der Physics Engine nach vorne getrieben. Die Spielwelt besteht aus tunnelartigen Objekten, deren Begrenzungen unsichtbare Wände darstellen. Somit kann sich der Spieler nur innerhalb des Tunnels bewegen. Der Tunnel selbst besteht aus mehreren Segmenten die jeweils hintereinander platziert sind. Hat der Spieler die erste Hälfte der verfügbaren Segmente passiert wird die zweite Hälfte inklusive des Spielers zurück zum Start befördert und dann einzeln wieder nach oben geschoben sobald der Spieler das entsprechende Segment verlassen hat. Somit bleibt der Spieler und die Spielwelt immer an etwa derselben Stelle und Probleme mit der Sichtweite oder Lichtberechnung werden vermieden.
Die einzelnen Segmente werden jedes Mal wenn der Spieler sie verlässt zufällig mit den festgelegten Objekten befüllt. Die Segmente werden dabei in einfache Quadrate zerlegt auf denen sich jeweils maximal ein Objekt befinden kann. Dabei haben die verschiedenen Objekte (Blütenstaub, Bäume, Regentropfen, etc.) einen Wahrscheinlichkeitswert mit dem sie auf einem Quadrat platziert werden.
Die Aufgabe des Spielers ist es nun Bäumen, Pilzen und Regen auszuweichen bzw. Blütenstaub einzusammeln um wieder Zeit dazu zubekommen. Bei Kollision mit Regentropfen wird dem Spieler Leben abgezogen. Schafft man es ein Level für eine gewisse Zeit zu überleben, ohne dass die Zeit abläuft so kommt man in das nächste Level. Im nächsten Level bekommt man einen gewissen Bonusbetrag zur Zeit hinzu welcher allerdings pro Level immer weniger wird (die Schwierigkeit steigt). Hat man verloren startet man durch Betätigen der Enter-Taste wieder von vorne.

3D Geometry
Um komplexe geometrische Objekte ins Spiel zu laden wird die „Assimp“ Bibliothek im Projekt verwendet. Damit erhalten wir auch Zugriff auf alle benötigten Daten, wie Vertices, Indices und Materialeigenschaften, um die Modelle zu rendern.
Verwendete Objekte sind selbst erstellte Low-Poly Objekte wie Bäume und Pilze. Diese wurden in Blender erstellt. Für unseren Spielcharakter verwendeten wir ein Schmetterling-Model.

Win/Loose Condition
Win Condition: Level-Up wenn jetziges Level für eine gewisse Zeit lang überlebt wurde.
Loose Conditions: Zeit läuft ab. Alle Leben durch Regentropfen verloren.

Intuitive Controls
Es wird eine Steuerung verwendet welche, der Tastenbelegung nach, in der Spieleindustrie zu einem inoffiziellen Standard geworden ist - WASD + Maus. Sowohl Tastatur- als auch Maus Input sind frameunabhängig.

Zusätzlich gibt es folgende Tastenbelegung:

ESC	Shutdown
F2	Toggle Wireframe
F3	Toggle FPS
F4	Reload Config
F6	Toggle HUD
ENTER	Restart

Es gibt Parameter in der Config welche einen Neustart erfordern und auch als solche ausgewiesen sind.

Intuitive Camera
Hierbei handelt es sich um eine 3rd-Person Kamera. Der Abstand kann in einem vorgegebenen Bereich mit dem Mausrad gewählt werden und ermöglicht somit auch das Spielen in 1st-Person. Da der Spieler in eine vorgegebene Richtung gedrängt wird ist die Rotation sowohl horizontal als auch vertikal beschränkt (einfach umzudrehen wird damit unmöglich). Die View-Projectionmatrix wird gegebenenfalls vom Shader vor dem Rendern aller angehängten Modelle eingeholt und gesetzt.

Textures
Die Texture-Koordinaten werden beim laden der Objekte verarbeitet. Bevor ein Mesh gerendert wird, wird die richtige Texture gebunden. Der Boden wurde mit einer einfachen Erd-Textur bedeckt.

Moving Objects
Der Spielcharakter wird mit der Kamera, die sich leicht versetzt hinter diesem befindet, durch die Spielwelt bewegt. Weitere bewegliche Objekte sind die Regentropfen, die sich mithilfe der Physics-Engine Richtung Boden bewegen. Framerate Independence wird in der Game-Loop implementiert.

Beleuchtung
Für die Beleuchtung der Szene verwenden wir ein Directional Light und das Phong Modell.

Adjustable Parameters

Dem Spiel liegt eine Datei “Config.txt” bei. Einige Parameter benötigen einen Neustart, die meisten können auch mit F4 geladen werden. Neben den verlangten Einstellungen gibt es noch ein paar weitere Parameter. Hier sind die interessantesten wohl die Sensitivity der Maus, die Shadowmap Resolution und die Difficulty.

Features

Physics Engine

“Physx” wurde mithilfe der Tutorials im Tuwel Forum in das Projekt eingebunden. Dabei verwenden wir fast nur dynamische Spielobjekte, da diese im Spiel ständig verschoben werden müssen. Zur Bewegung des Spielers verwenden wir einen Character Controller. Physx wird in unserem Projekt außerdem noch für die Collision Detection verwendet. Dazu wurden die Meshes hinreichend unterteilt damit das Convex Mesh von Physx als Collision Shape zum jeweiligen Model passt.

HUD

Unser Spiel verwendet ein Heads Up Display, welches dem Spieler Informationen über den Spielstatus geben soll. Links oben befindet sich die Lebensanzeige, in der Mitte die ablaufende Spielzeit und rechts die FPS, die mit F3 aus/ein geschaltet werden können. Das HUD selbst kann mit F6 aus/ein geschaltet werden. Geht dem Spieler die Zeit oder Leben aus so wird der jeweilige Lose Screen angezeigt. Für das Rendern von Text wurde die Freetype Library verwendet. Für eine ausgewählte Schriftart wird eine Bitmap erstellt aus dieser die einzelnen Buchstaben extrahiert werden können.

Effekte

Lighting - Shadow Maps

Vor dem Postprocessing und dem eigentlichen Zeichnen des Spiels werden die Tiefenwerte der Szene (aus Sicht der Lichtquelle) in einer Textur abgelegt (durch Nutzung eines anderen Framebuffers, siehe auch Post Processing). Beim Zeichnen des Spiels wird der aktuelle Punkt im Shader mithilfe der anderen View-Projectionmatrix zur Lichtquelle hin berechnet. Ist der Tiefenwert gleich haben wir die Stelle gefunden welche von der Lichtquelle angestrahlt wird. Ist der Tiefenwert größer (weiter weg) als der näheste Wert muss es sich um einen Punkt im Schatten handeln. Punkte im Schatten erhalten nur Ambient Light. Seiteneffekte wie “Shadow Acne” oder “Peter Panning” wurden behandelt. Die Shadowmap wird außerdem in einem 3x3 Feld um den aktuellen Punkt herum verwendet um die Schattenqualität zu erhöhen (PCF).
Die Auflösung der Shadowmap ist im Config File einstellbar.

Advanced Modeling - Cpu Particle System

An der Decke jedes Segments hängt ein Particle System welches für die Regentropfen zuständig ist. Die Tropfen bewegen sich dabei in Richtung des Bodens und sind in der Physics Engine da wir damit kollidieren müssen und die Collision Detection über Physx gelöst ist.
Der Startpunkt eines Partikels wird zufällig in jedem Update-Zyklus gewählt. Wenn ein Regentropfen eine Kollision hatte oder seine Lebenszeit vorüber ist wird er aus dem Spiel entfernt.

Die Partikel werden mit Instancing Calls gezeichnet. Wir sind dazu übergegangen auch alles andere instanced zu rendern und haben die FPS so um einiges nach oben gedrückt.

Animation - Hierarchical Animation
Dieser Effekt wird dazu verwendet um die Flügeln des Schmetterlings zu animieren. Damit sich die Flügeln schön um den Körper bewegen befindet sich im Körper des Schmetterlings jeweils ein kleines Objekt, welches das ParentObjekt des jeweiligen Flügels darstellt. Diese Objekte werden bis zu einem definierten Winkel um eine fixe Schrittgröße rotiert und wenn der maximale Winkel erreicht wird ändert sich die Rotationsrichtung. Die Animation ist Frame unabhängig.

Shading - Cell Shading
Wird nur in einem der Shader verwendet und bricht die Koeffizienten der Lichtberechnung im Phong Modell auf “Shades” herunter. Das Ergebnis ist ein “Toon-Look” einiger Objekte im Spiel.

Post Processing - Contours via edge detection
Bevor das Spiel gezeichnet wird aktivieren wir einen anderen Framebuffer. Somit haben wir die fertige Ausgabe für Farb- und Tiefenwerte jeweils als Textur und verwenden diese in einem Postprocessing Shader. Hier wird eine Edge Detection mit Sobelfiltern horizontal und vertikal durchgeführt - und zwar auf den Tiefenwerten (die perspektivische Projektion wurde rückgerechnet um die Tiefenwerte auch ordentlich verwenden zu können).
Das Ergebnis wird auf einem Quad mit sechs Vertices auf dem originalen Framebuffer gezeichnet und enthält die schwarzen Outlines um die Objekte herum.

Quellen:

[1] https://learnopengl.com/Model-Loading/Assimp
[2] https://clara.io/view/667dc9ee-a9f8-4ecd-bf1b-c7f0d300a149
[3] https://learnopengl.com/In-Practice/Text-Rendering
[4] https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping
[5]https://computergraphics.stackexchange.com/questions/3646/opengl-glsl-sobel-edge-detection-filter