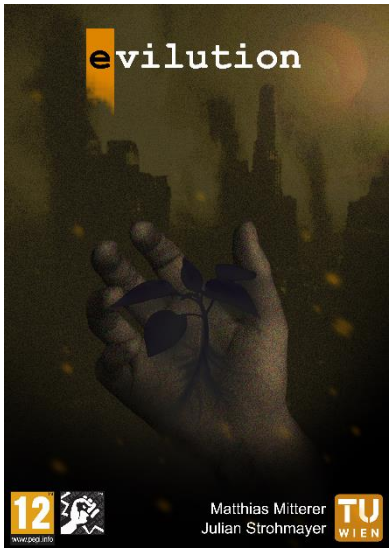
evilution – Dokumentation

(Submission 2.0)

[Steuerung]

W,A,S,D	**Spielersteuerung**
Maus	**Kamerasteuerung**
Linke Maustaste	**Speer abfeuern**
F1	**Sound Ein/Aus**
F2	**FPS Ein/Aus**
F3	**Wireframe Modus Ein/Aus**
F4	**Textur Sampling Qualität**
F5	**Mip Mapping Qualität**
F6	**God Mode Ein/Aus**
F7	**Shadow Mapping Ein/Aus**
F8	**Frustum Culling Ein/Aus**
F9	**Transparenz Ein/Aus**
ESC	**Spiel beenden**



Gameplay

Kurz gesagt ist das Ziel des Spielers, so lange zu überleben wie nur irgendwie möglich. Verhindert wird das durch die Feinde im Spiel (Bären), welche sich dem Spieler von allen Seiten nähern und versuchen ihm Schaden zuzufügen. Wie der derzeitige Stand der Gesundheit ist, kann am linken unteren Rand des Fensterns, an der Lebensanzeige abgelesen werden. Schafft es ein Feind sehr nahe an den Spieler heranzukommen, ohne vorher getötet zu werden, dann erleidet der Spieler Schaden. Verlorenes Leben wird wieder durch die Lebensanzeige dargestellt. Sinkt das Leben des Spielers auf null dann ist das Spiel vorbei. Um das zu verhindern, kann der Spieler durch klicken der linken Maustaste Speere abfeuern. Trifft der Spieler mit einem Speer einen Feind, dann stirbt dieser und es wird sofort ein neuer Freund irgendwo auf dem Spielfeld erzeugt. Der neu erzeugte Feind ist seinen Vorgängern aber überlegen. Wenn ein neuer Feind erzeugt wird, erhält dieser wesentlich bessere Fähigkeiten. Das bedeutet höhere Bewegungsgeschwindigkeit und erhöhter Schaden. Dadurch steigt der Schwierigkeitsgrad kontinuierlich an und es wird ab einem gewissen Zeitpunkt kaum mehr möglich sein zu überleben.

Frei bewegliche Kamera

Die Kamera Bewegung wird über die Methode *Game::handleInput(float time_delta)* abgewickelt. Beim Drücken der W, A, S, D Tasten, oder beim Verschieben der Maus wird die Position aktualisiert und die Projektionsmatrix und die View Matrix neu berechnet. Diese stehen dann für die Berechnung der MVP Matrix zur Verfügung. Im normalen Spielmodus kann sich der Spieler nur auf dem Terrain fortbewegen. Wird jedoch die F6 Taste gedrückt, dann befindet man sich im „God Mode“ und kann über das Terrain fliegen. Diese Funktionalität ist nicht Teil des eigentlichen Gameplays, sie dient nur als Debugging Tool.

Frei bewegliche Objekte

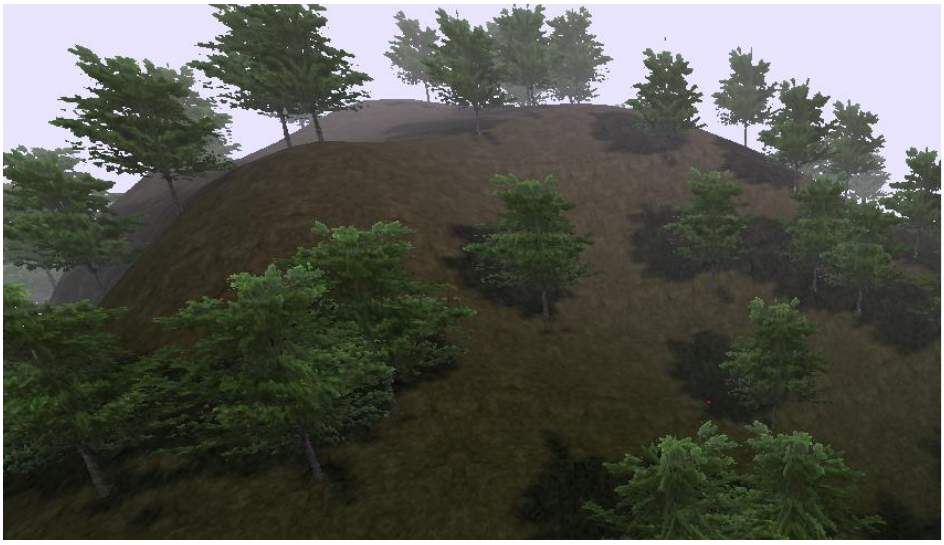
Jedes Objekt in der Szene ist Unter Typ von „SceneObject“ und besitzt daher die Methode *SceneObject::update(float time_delta)*. Diese Methode manipuliert die Model Matrix des SceneObjects. Für jedes bewegliche Objekt wird im GameLoop in jedem Frame die Methode update(float time_delta) aufgerufen. Frei bewegliche Objekte sind unter anderem die Bären, der Speer den der Spieler abschießen kann, sowie auch der Adler den man direkt am Startpunkt beobachten kann.



Textur Mapping

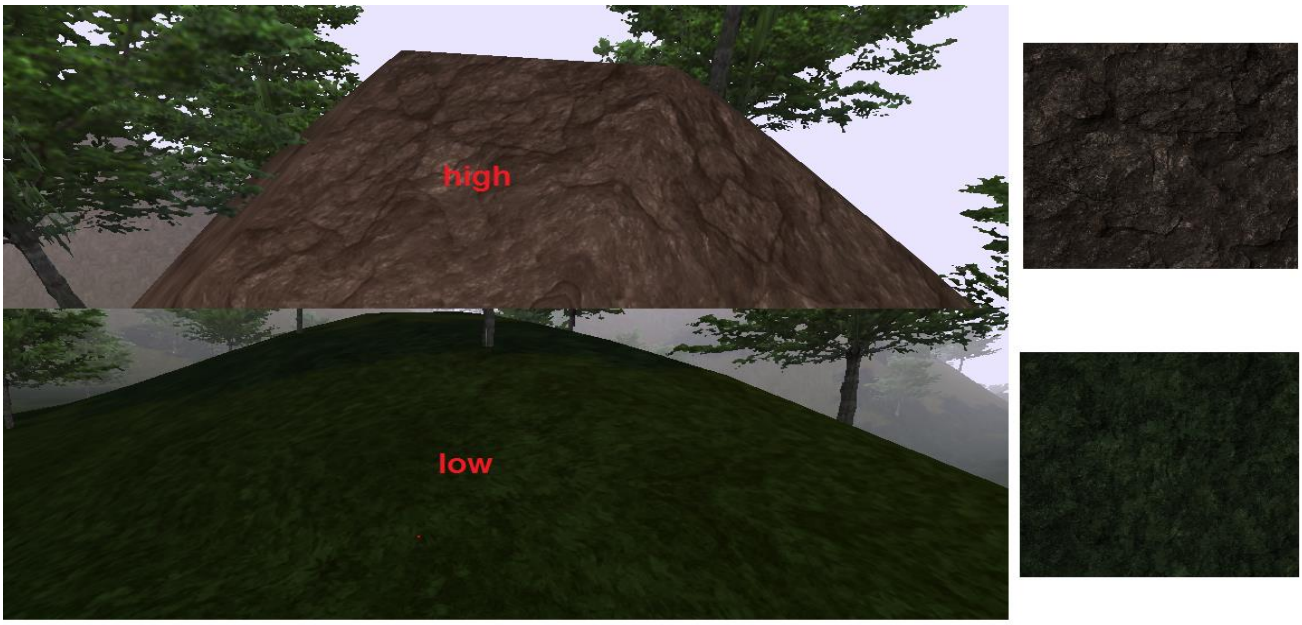
[Basic]

Das Textur Mapping wird von dem Shader (basic.vert / basic.frag) implementiert. Dieser Shader unterstützt RGBA und wird für Objekte wie Bäume, Bären, Speer und das Lagerfeuer verwendet. Um weiße Ränder bei Texturen mit Alpha Kanal zu vermeiden, werden Pixel verworfen wenn sie einen gewissen Alpha Channel Schwellwert unterschreiten. Hier ein Beispiel dieses Shaders.




[Terrain]

Die Texturierung des Terrains wird durch den Shader (terrain.vert/ terrain.frag). Dieser Shader unterstützt auch RGBA. Die Besonderheit dieses Shader liegt darin, dass er zwei Texturen über die Oberfläche des Objektes interpoliert. Als Interpolationsparameter dient dabei die Höhe (Wert der Y-Achse). Ein Punkt mit einem sehr geringen Höhenwert wird ausschließlich mit der sogenannten „lowTexture“ texturiert. Ein Punkt der sehr weit oben liegt wird mit der „highTexture“ texturiert werden. Punkte die dazwischen liegen werden wie schon angesprochen interpoliert. Zusätzlich zur Interpolation zwischen den Texturen, wird ebenfalls eine Grundfarbe und die Helligkeit über die Höhe variiert um das Terrain etwas natürlicher wirken zu lassen. Im unten angeführten Bild sehen Sie ein Extrembeispiel dieser Interpolation.



[Wasser]

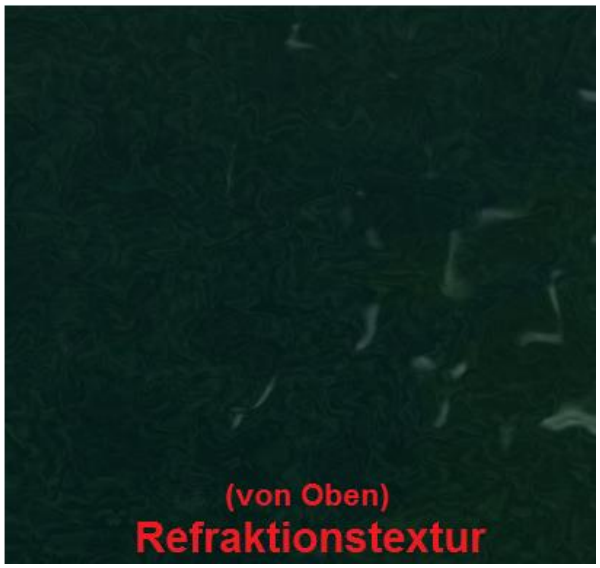
Auch das Wasser besitzt einen eigenen Shader (newWater.vert / newWater.frag). Dieser Shader benutzt ebenfalls Texturen um die Wasseroberfläche darzustellen. Für die Wellenbewegung wird eine DUDV-Textur verwendet und für das Normal Mapping eine Normal-Textur. Außerdem Werden Reflexions- und Refraktionstexturen auf das Wasser gerändert und nach fresnel interpoliert. Ein Solches Beispiel mit den verwendeten Texturen finden Sie im unten angeführten Bild.

Beleuchtung und Materialien

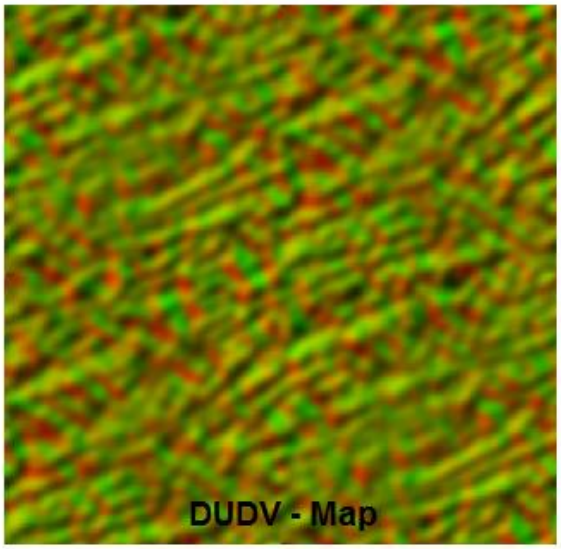
Als einzige statische Lichtquelle im Spiel fungiert eine globale Sonne. Die Beleuchtung kann je nach Material in den dafür implementierten Shadern variiert werden. Beispielweise wird das Terrain ausschließlich diffus beleuchtet. Das Wasser wiederum besitzt neben diffuser Beleuchtung außerdem spekulare Beleuchtung, wodurch je nach Position des Spielers Glanzpunkte auf die Wasseroberfläche sichtbar werden.

Effekte

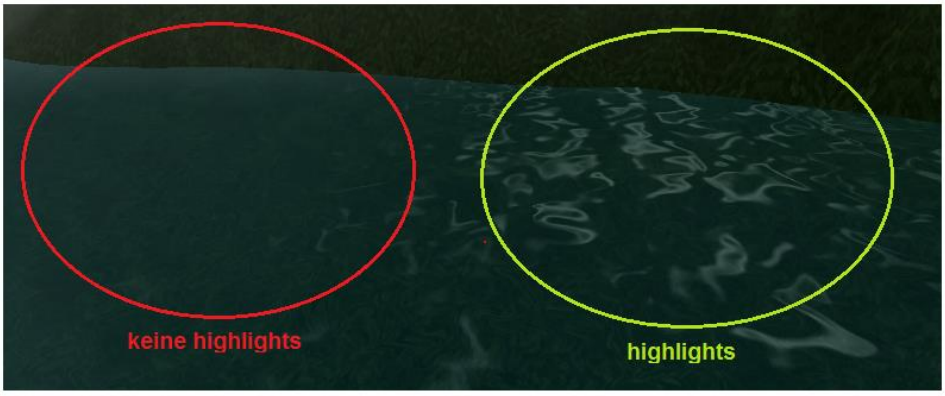
[Water | 2.0P] (Fesnel-Shading, Normal Mapping, Moving Waves, Reflection, Refraction)
Der Erste Effekt der in unserem Spiel implementiert wurde, ist eine Wasseroberfläche mit Reflektion, Refraktion, Wellenbewegung und Normal Mapping. Der Dafür zuständige Shader ist (newWater.vert/newWater.frag). Die Textur für die Reflektion wird erzeugt indem die Kamera in jedem Frame, um die doppelte Distanz zwischen Spieler und Wasseroberfläche, nach unten bewegt wird. Anschließend wird der Pitch der Kamera invertiert, damit die Kamera von unten auf die Szene schaut. Die Szene wird dann in ein FBO gerendert. Für die Refraktionstextur wird ohne Kamerabewegung in ein FBO gerendert. Diese beiden Texturen werden dann von Textur- in Weltkoordinaten umgerechnet und nach dem Fresnel Faktor interpoliert.

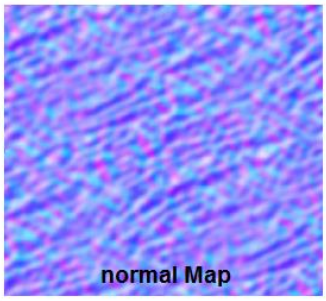


Für die Wellenbewegung haben wir eine DUDV-Map verwendet, welche die Texturkoordinaten verzerrt. Dadurch entsteht der Effekt, dass sich die eigentlich statische Oberfläche verschiebt. Das Aussehen der Wellen kann in den Shadern durch das Attribut „tiling“ und „moveFactor“ angepasst werden.



Für die spekularen Highlights haben wir eine Normal Map verwendet, welche die eigentlich senkrecht nach oben zeigenden Normalvektoren der Wasseroberfläche manipuliert. Dadurch entsteht je nach Blickrichtung der Glanzeffekt auf den Wellen.

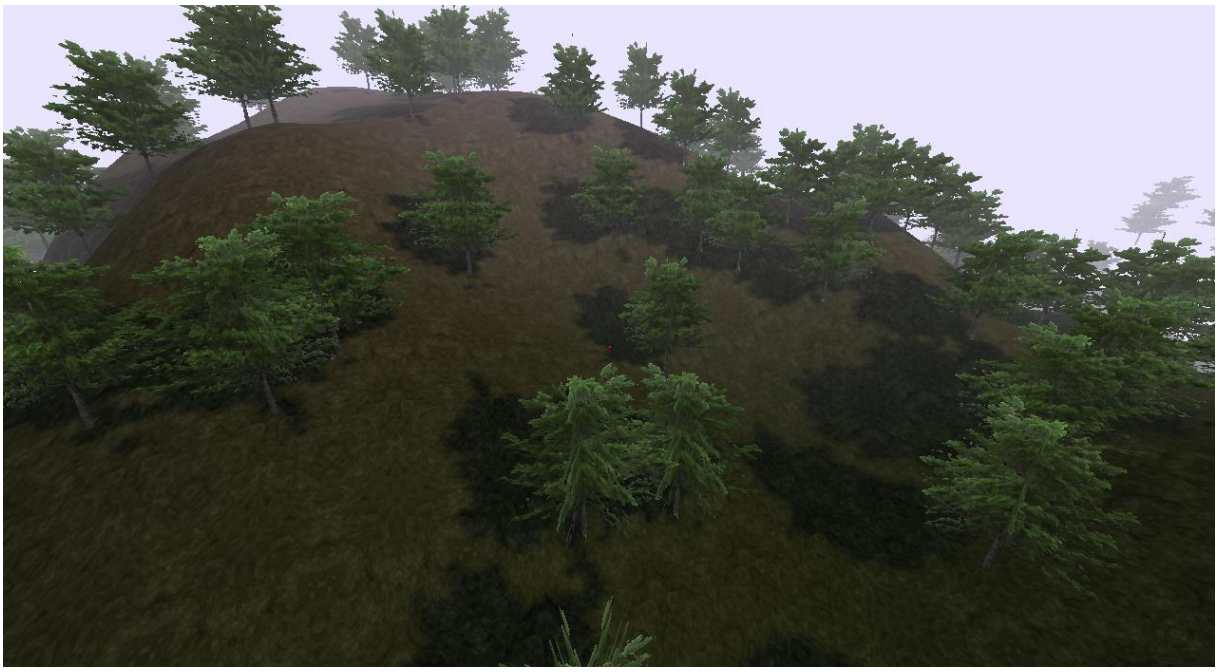




Hier ein Beispiel für die Kombination aller Effekte.

[Shadow Maps | 1.5P] (with PCF)

Die Schatten der Bäume auf dem Terrain werden mittels Shadow Mapping geworfen. Dabei wird die Szene als Zwischenschritt aus Sicht der Lichtquelle in eine Textur gerendert. Da nicht alle Informationen benötigt werden, werden nur die Tiefen-Informationen der nicht-transparenten Pixel gerendert. Im zweiten Schritt wird die Szene auf den Bildschirm gerendert. Dabei wird verglichen, ob die Tiefe des gerenderten Pixels größer ist, als die Tiefe in der Shadow Map. Wenn das der Fall ist, dann wird das Licht durch ein Objekt blockiert und es muss ein Schatten geworfen werden. Um Störartefakte durch Shadow Acne und Aliasing zu vermeiden, wurde ein Bias-Wert und PCF eingebaut. Dabei werden die Kanten der Schatten mit der Untergrundfarbe interpoliert, um die Kanten der Schatten zu glätten und natürlicher aussehen zu lassen.



[GPU Particle System | 1.0P] (Transform Feedback, Instancing)

Für das Lagerfeuer wurde ein Partikelsystem verwendet. Zu diesem Zweck wurde ein Partikel-Objekt mit folgenden Eigenschaften definiert: Position, Beschleunigung, Farbe, Lebenszeit, Größe und Typ. Der entscheidet, ob das Partikel eine Vorlage oder ein zu renderndes Partikel ist. Anschließend werden in jedem Frame zwei Shader-Durchläufe ausgeführt:

- Update: Die Partikel werden anhand der übergebenen Parameter bewegt. Wenn ein Partikel die Lebensdauer überschritten hat, wird ein neues erzeugt. Da die neu-Erzeugung der Partikel die Erzeugung von Geometrie voraussetzt, werden dafür Geometry-Shader verwendet, die genau für solche Aufgaben vorgesehen sind. Die neu berechneten Werte werden vom Shader mittels Transform Feedback zurück geliefert.

- Rendern: im zweiten Schritt werden die Partikel auf den Bildschirm gerendert. Für jeden Partikel wird im Shader ein Quadrat erzeugt und texturiert.

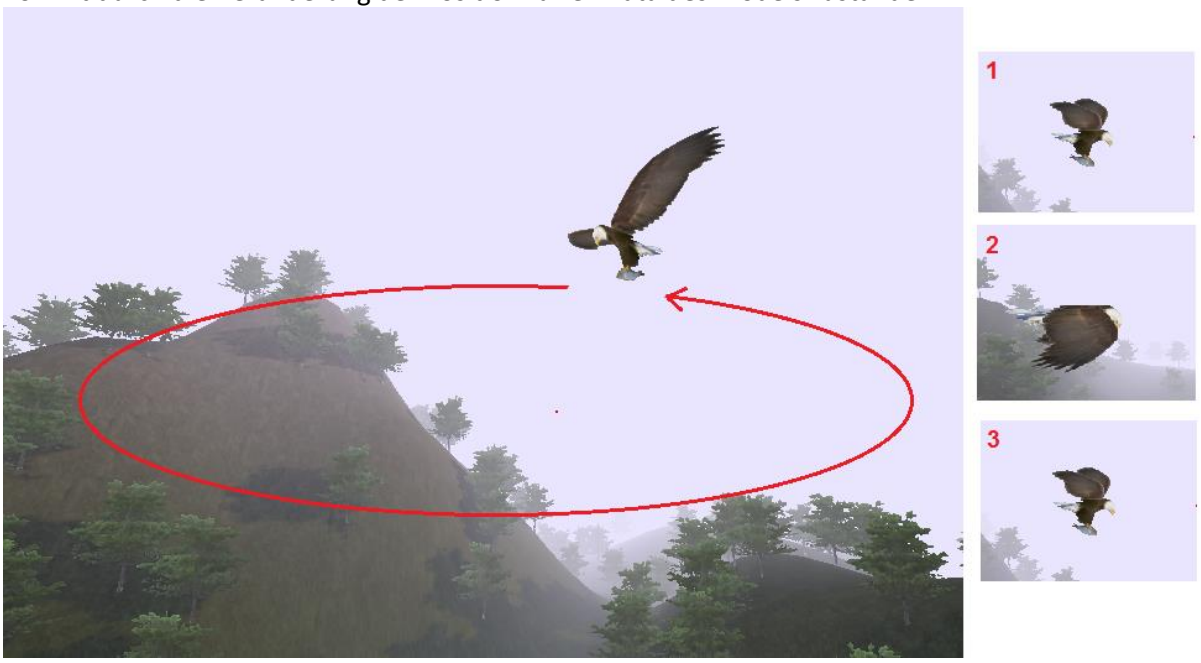
Frustum Culling

Da Objekte in der Ferne wegen dem Nebel-Effekt nicht sichtbar sind, müssen diese auch nicht gerendert werden. Das wird dadurch erreicht, dass für alle Objekte des Terrain- und Wasser-Rasters und der Pflanzen die Distanz zur Kamera berechnet wird. Liegt diese über einem Schwellwert, wird das Objekt nicht gerendert. Durch diese Optimierung läuft das Spiel am Abgaberechner mit ca. 100 fps mehr.

Animation

[Adler]

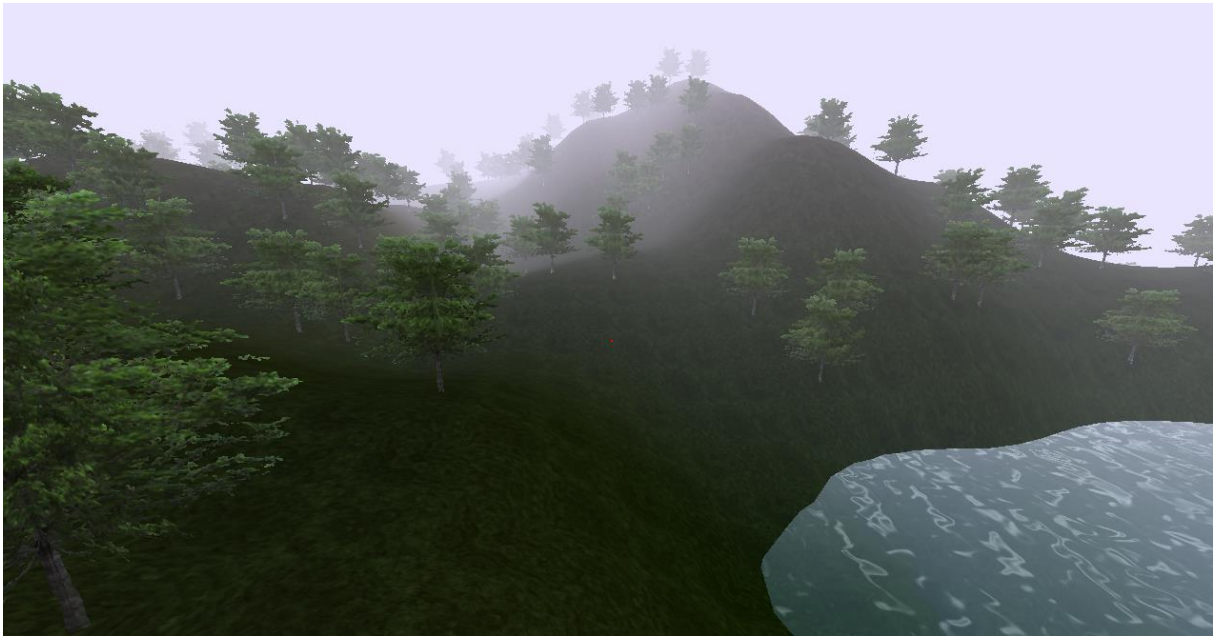
Als Animation haben wird einen Fischadler eingebaut, welcher in unmittelbarer Nähe der Startposition über einem kleinen Teich kreist. Dabei schlägt er mit den Flügeln auf und ab. Die Animation kommt durch die Veränderung der PositionBuffer-Data des Models zustande.



Besondere Features

[Terrain Generierung mit Perlin Noise]

Das Terrain in unserem Spiel wurde nicht händisch modelliert. Für die Erzeugung wurde Perlin Rauschen verwendet, um für jeden Punkt im Raum einen fest definierten Höhenwert zu berechnen. Durch die Funktionseigenschaft verändert sich das Terrain bei Neustart des Spiels nicht. Die Höhenwerte sind immer gleich und im gesamten Reellen Raum definiert, solange nicht die Funktionsparameter verändert werden. Die Verwendung von Perlin Noise gibt uns die Möglichkeit das Terrain beliebig groß zu gestalten, solange es zu keiner Hardwarelimitation kommt. Derzeit wird ein eher kleines Terrain 256x256 verwendet.



[Pflanzen Generator]

Für die automatisierte Bepflanzung eines beliebig skalierbaren Terrains haben wir einen Plant Generator entwickelt, welcher jede Form von SceneObjects auf dem Terrain verteilen kann. Um die Szene natürlich wirken zu lassen, werden einige Objekte willkürlich rotiert. Beim Erstellen eines neuen Generators kann ein Parameter übergeben werden, welcher darüber entscheidet, wie viele Objekte auf dem Terrain verteilt werden sollen. Außerdem kann ein „Random Seed“ angegeben werden, damit die Position, an der Objekte platziert werden, immer variiert.

[Nebel Effekt]

Wie man schon in allen vorherigen Bildern erkennen kann, haben wir in allen Shadern einen Nebel Effekt implementiert um die Szene natürlicher wirken zu lassen. Wir interpolieren dabei die Pixelfarbe mit der Hintergrundfarbe der Szene `0.92f, 0.90f, 1.0f, 1.0f`. Als Interpolationswert wird die Distanz zur Kamera verwendet.

[Sound]

Während des gesamten Spiels wird eine Sounddatei geloopt, welche für ein natürliches Wald-Ambiente sorgt. Der Sound kann mit der F1-Taste Ein/Aus geschaltet werden.