Beuth Hochschule für Technik Berlin

- Fachbereich Informatik und Medien -

Powerpong

ImModul

Computer Graphics and Effects

Bei Prof. Dr. Henrik Tramberend

vorgelegt von

Lars George [770438] – lars.george@gmx.net

Chris Krauß [770822] – chris@redvirus.de

Bearbeitungszeitraum: Wintersemester 2010/11

Inhaltsverzeichnis

1 Einleitung ... 1
2 Aufgabenbeschreibung ... 2
3 Spielidee ... 3
4 Features ... 5
4.1 Wii Controle ... 5
4.2 Physik mit jBullet ... 5
4.3 Portal ... 5
4.4 Multiplayer ... 6
4.5 Sound ... 6
4.6 Ghost-Train ... 6
4.7 Die Waffen ... 7
4.7.1 Peagun ... 7
4.7.2 Gluegun ... 7
4.7.3 Object activator ... 8
4.8 Partikelsystem ... 9
5 Verwendete Shader ... 11
5.1 Ball ... 11
5.2 Energy bar ... 12
5.3 Portal ... 12
5.4 Point Sprites ... 13
5.5 Environment Mapping ... 14
5.6 Parallax Mapping ... 15
5.7 Shadow Mapping ... 15
6 Postprocessing ... 16
6.1 Color inversion ... 16
6.2 Bloom ... 17
6.3 Nightvision ... 17
6.4 Pixelation ... 18
6.5 Depth of Field ... 19
7 Installationsanleitung ... 20
Abbildungsverzeichnis ... 21
Tabellenverzeichnis ... 21
Listingverzeichnis ... 21

1 Einleitung

In diesem Dokument wird die Projektarbeit von Lars George und Chris Krauß im Modul Computer Graphics and Effects an der Beuth Hochschule für Technik Berlin im Wintersemester 2010/2011 beschrieben. Es wird zunächst die Aufgabenstellung von Prof. Dr. Tramberend gezeigt. Die eigene Spielidee wird anschließend in einem eigenen Kapitel dargelegt. Die weiteren Kapitel beschrieben die jeweiligen umgesetzten Bereiche des Prototyps und erklären die eingesetzten Techniken. Am Schluss wird eine Anleitung zur Benutzung des Prototyps gegeben.

2 Aufgabenbeschreibung

Im Modul Computer Graphics and Effects sollte ein interaktives 3D Spiel erstellt werden. Das Ziel ist dabei einen spielbaren Prototyp zu erstellen, welcher vor allem technisch anspruchsvolle Shader-Effekte nutzt.

Folgende Phasen wurden dabei vorgegeben und eingehalten:

1. Konzeption Spielidee
2. Implementierung Spielgerüst
3. Modelle und Effekte
4. Abgabe und Präsentation

Zum Teil der Abgabe gehört auch dieses Dokument. Zusätzlich wurde ein Film aufgenommen, welcher die jeweiligen Effekten in bewegten Bildern vorstellt.

Die ursprüngliche Kriterien-Gewichtung wurde wie folgt festgelegt:

Umsetzung der Idee	20%
Integration der Shadertechniken	20%
Integration Holodeck	10%
Technische Schwierigkeit	30%
Code-Qualität	10%
Dokumentation & Video	10%

Eine Integration für das Holodeck wurde jedoch aus der Bewertung entfernt. Die Technik konnte nach einem Softwareupdate nicht rechtzeitig stabil in Betrieb genommen werden.

3 Spielidee

Die Idee basiert auf dem Spiele Klassiker Pong. Beim ursprünglichen Pong steuert der Spieler einen „Paddel“ und versucht damit den im Level rumfliegenden „Ball“ abzuwehren. Gerät der Ball hinter eine Grundlinie, so erhält das jeweils andere Team einen Punkt.

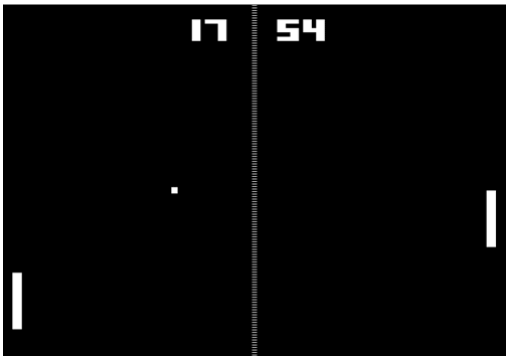


Abbildung 3.1: Spiele Klassiker Pong

Powerpong erweitert dieses Spielprinzip. Zunächst wird die Ansicht in eine dreidimensionale erweitert. Die Sicht ist eine First-Person-Cam. Das Steuern der Paddel wird auf beiden Seiten von einer künstlichen Intelligenz übernommen. Der Spieler erscheint als ein Avatar auf dem Spielfeld und kann den Verlauf des Balls beeinflussen, um so seinem Team Vorteile zu verschaffen und zu punkten. Hierfür bekommt der Spieler ein Arsenal von Waffen mit verschiedenen Funktionen. In Tabelle 1 sind die geplanten Waffen mit einer Beschreibung aufgelistet.

Tabelle 3-1: Waffenarsenal des Spielers in Powerpong

Name	**Beschreibung**
Gluegun	Es wird eine Fläche erzeugt, welche den Ball verlangsamt, wenn er darüber kommt.
Peagun	Kleine Kügelchen werden verschossen und lenken den Ball ab.
ObjectActivator	Erzeugt ein neues „Ghost“- Objekt in der Welt.

Das Abfeuern einer Waffe kostet Energie. Der Spieler regeneriert kontinuierlich Energie. Es muss entschieden werden, für welche Effekte die Energie aktuell verwendet werden soll, da das langsame Regenerieren ein dauerhaftes Benutzen der Waffen unmöglich macht.

Die folgende Abbildung zeigt eine Momentaufnahme des umgesetzten Prototyps.

Abbildung 3.2: Darstellung des umgesetzten Prorotyps zu Powerpong

4 Features

In diesem Kapitel werden die umgesetzten Features bei Powerpong gezeigt. Die eingesetzten Shader werden im nächsten Kapitel extra behandelt. Bis auf eine Waffe (Accellometa) wurden alle im Exposé beschriebenen Eigenschaften umgesetzt. Darüber hinaus wurden diverse weitere Möglichkeiten und Effekte realisiert. Diese werden im Folgenden Vorgestellt.

4.1 Wii Controle

Zur Steuerung kann neben der Tastatur und Maus ebenfalls die Wiimote mit angeschlossenem Nunchuck verwendet werden. Ursprünglich war dies angedacht, um das Spiel an das Holodeck an zu passen. Bei Tests konnte jedoch gezeigt werden, dass gerade für Leute welche nicht intensiv Computerspiele spielen, das Steuern mit der Wiimote einfacher und intuitiver ausfiel.

4.2 Physik mit jBullet

Bei der Berechnung von physikalischen Eigenschaften wird jBullet verwendet. Alle Objekte besitzen eine optische Repräsentation und ein entsprechendes physikalisches Pendent. Dies ermöglicht die Beeinflussung aller Objekte untereinander, wie beispielsweise das Schießen einer Kugel in eine Kistenmenge.

Ein Spieler besitzt ebenfalls ein physikalisches Objekt, damit er beispielsweise vom Spielball weggeschoben werden kann. Die Möglichkeit zu springen und sich zu bewegen wird durch das Anlegen einer zentralen Kraft im Masseschwerpunkt des Spielerobjekts in die jeweilige Richtung erreicht.

Die Waffen-Effekte profitieren alle von der Verwendung der Physik-Engine. Ein Peaball ist lediglich eine Kugel mit einer gewissen Masse und weiteren physikalischen Eigenschaften. Die vom Object activator erzeugten Kisten funktionieren ähnlich, sie erhalten jedoch durch eine andere Rendertechnik eine andere spielbeeinflussende Möglichkeit. Bei der Gluegun werden Kollisionen mit dem Ball überprüft und gegebenenfalls dessen Bewegungsvektor verkleinert.

4.3 Portal

Auf dem Spielfeld von Powerpong befinden sich zwei verbundene Portale. Betritt der Spieler / Spielball eine Seite, so kommen diese auf der jeweils anderen Seite wieder heraus. Dieser Riss im Raum ist jedoch nicht ohne Folgen für die Powerpong-Welt. Weiteres wird im Abschnitt 4.6 Ghost-Train erklärt.

Bei der Visualisierung wurde eine veränderte Version des Spiegelshaders eingesetzt. Näheres kann im fünften Kapitel nachgelesen werden.

4.4 Multiplayer

Powerpong kann mit mehr als einem Spieler gespielt werden. Hierfür wurde ein eigenes Client/Server System entwickelt. Grundlage hierfür bildet das MINA [1]Framework von Apache. Der Austausch von Informationen geschieht Nachrichtenbasiert. Auf dem Serverprogramm wird die virtuelle Welt stetig weiterberechnet. Clients senden ihre Eingaben, beispielsweise der Wunsch sich nach vorne zu bewegen, an den Server, welcher dies umsetzt und allen verbundenen Clientprogrammen über die neue Position informiert.

Die Möglichkeit mit mehreren Spielern gleichzeitig zu spielen steigert den Spielspaß enorm. Spieler können sich gegenseitig beschießen und versuchen für ihr eigenes Team jeweils den Ball vorteilhaft zu beeinflussen.

Um die Last des Servers zu senken und weniger Datenverkehr zu verursachen wurden Optimierungstechniken angewandt. So werden beispielsweise beim Schießen einer Waffe lediglich die Start und Endpunkte vom Server übermittelt. Die Positionsberechnungen dazwischen werden lokal vorgenommen und entsprechend angezeigt.

4.5 Sound

Zum Abspielen für Musik und Soundeffekte wurde eine Audiomanager Klasse implementiert. Diese unterscheidet zwischen Musik in MP3-Form und Soundeffekten, welche als WAV-Datei vorliegen müssen. Die Soundeffekt-Dateien werden beim Start vorgeladen, damit diese schnell in der jeweiligen Situation abgespielt werden können. Musikdateien werden nicht mitgeliefert. Hierfür gibt es den mp3 Ordner im Projekt. Benutzer können dort ihre eigenen Musikwünsche ablegen und im Spiel anhören. Hierfür können die Lieder weitergeschaltet (.) und gestoppt (,) werden.

4.6 Ghost-Train

In unregelmäßigen Abständen fährt ein Zug von einem Portal zum Anderen. Diese außerdimensionale Bahnlinie ist nur mit dem Nachtsichtgerät erkennbar (aktivier bar mit der Taste „N“). Der Zug verhindert kurzzeitig das Herüberspielen des Balles. Damit man über das Erscheinen informiert wird, ertönen entsprechenden Sounds am Anfang der jeweiligen Fahrt. Der Zug unterteilt die Spielfläche kurzweilig in der Mitte und verhindert einen Seitenwechsel.

[1] http://mina.apache.org/

4.7 Die Waffen

Alle Waffen besitzen eine Energieanzeige, welche aufzeigt ob ein weiterer Schuss möglich ist. Dies wird im Kapitel über die eingesetzten Shadertechniken näher erläutert. Im Rahmen des Prototypen wurden drei Waffen umgesetzt.

4.7.1 Peagun

Mit der Peagun werden Kugeln verschossen, welche lediglich durch ihre Masse und Größe Objekte beeinflussen. Sie besitzen eine zeitliche Begrenzung und werden automatisch aus dem Szenengraphen und der physikalischen Welt entfernt, wenn ihr gesetzter Timer abgelaufen ist.

Abbildung 4.1: Die Peagun

4.7.2 Gluegun

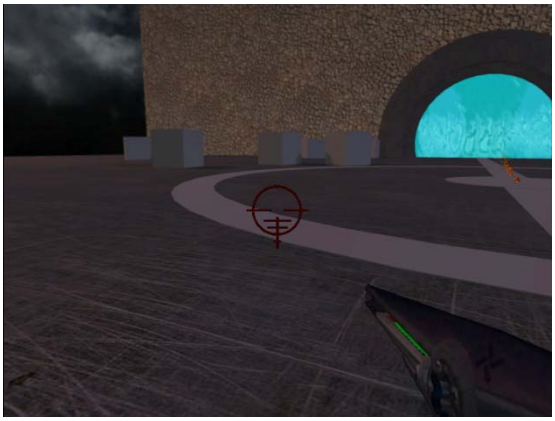


Abbildung 4.2: Die Gluegun

Mit der Gluegun wird zunächst eine Kugel verschossen. An dem Ort wo diese Kugel den Boden berührt, wird anschließend eine grünliche Masse erzeugt. Die Masse hat ein eigenes Material und somit auch eigenen Shader bekommen. Dies wird im nächsten Kapitel unter dem Punkt Environment Mapping erläutert. Zusätzlich wird der grüne Farbwert hervorgehoben. Diese „Gluepaste“ bremst

den Spielball erheblich ab, wenn dieser mit ihr kollidiert (über sie rollt). Der dafür notwendige Quellcode wird im folgenden Listing gezeigt:

Listing 4.1: Auszug aus der Methode checkGluePaste in der Klasse PhysicManager

```
private void checkGluePaste(RigidBody bodyA, RigidBody bodyB, Game game)
{
      if (ballHitsObject(bodyB, bodyA, GluePaste.class)
          || ballHitsObject(bodyA, bodyB, GluePaste.class))
      {
            Ball b = null;
            if (bodyA.getMotionState().getClass().equals(Ball.class))
            {
                  b = (Ball) bodyA.getMotionState();
            }
            else
            {
                  b = (Ball) bodyB.getMotionState();
            }

            Vector3f force = new Vector3f();
            force = b.getRigidBody().getLinearVelocity(force);
            force.scale(GlueGun.movementStopper);

            b.getRigidBody().setLinearVelocity(force);
      }
      [...]
}
```

4.7.3 Object activator

Abbildung 4.3: Der Object activator

Durch den Object activator werden Kistenobjekte in der virtuellen Welt erzeugt. Diese sind auf eine gewisse Anzahl von Objekten auf dem Server begrenzt. Wird ein Objekt mehr erzeugt als die eingestellte Grenze ist, so wird das „älteste“ erzeugte Objekt für den neuen Schuss verwendet. Erzeugte Objekte blockieren sowohl den Spieler als auch den Spielball. Hiermit können Spieler

kleinere Barriere errichten. Sichtbar sind diese jedoch lediglich im Nachtsichtmodus (aktivierbar mit der Taste „N“).

4.8 Partikelsystem

Ein eigenes Partikelsystem wurde für verschiedene Zwecke umgesetzt. Das Feuer aus den Fackeln am Spielfeldrand, der Rauch aus den Waffen oder die Funken wenn der Ball eine Begrenzung trifft werden damit erzeugt. Viele Parameter zur Konfiguration machen es vielseitig einsetzbar. Jedes Partikel besitzt die folgenden Eigenschaften:

- Position
- Alter (Energie)
- Bewegungsrichtung
- Bewegungsgeschwindigkeit

Der Konstruktor für das Partikelsystem wurde wie im unten gezeigten Listing umgesetzt. Die jeweilige Erklärung der Parameter wird im Java-Quellcode gegeben:

Listing 4.2: Der Konstruktor des Partikelsystems

```
        /**
         *
         * @param position
         *            start position of the particles
         * @param mainDirection
         *            main flow direction of the particles
         * @param directionDistortion
         *            the x,y,z values of this vector define how the
         *            particles have scattered directions (around the main
         *            direction vector)
         * @param minSpeed
         *            min speed of particles
         * @param maxSpeed
         *            max speed of particles
         * @param numParticles
         *            how many particles should be generated max
         * @param emitRate
         *            how many particles should be spawnt per second
         * @param particleSize
         *            the size of a particle
         * @param durationTimeInSeconds
         *            how long should the particle system exist? 0 means
         *            infinite
         * @param continouslyEmitting
         *            should particles that disapeared be respawned?
         * @param energyDamping
         *            how should be energy of a particle be damped (values
         *            between 0 and 1). energy is used to calculate the
         *            visibility of a particle
         * @param startEnergy
         *            how much start energy do the particles have
         * @param spriteTextureFile
         *            the texture to use for the particles
         * @param gravity
         *            a gravity vector. this one does not need to direct to
         *            the ground. a vector to the "left" would result in
         *            something like wind
         */
public ParticleSystem(Vector3 position,
                      Vector3 mainDirection,
                      Vector3 directionDistortion,
                      float minSpeed,
                      float maxSpeed,
                      int numParticles,
                      double emitRate,
                      float particleSize,
                      double durationTimeInSeconds,
                      boolean continouslyEmitting,
                      double energyDamping,
                      float startEnergy,
                      String spriteTextureFile,
                      Vector3 gravity)
```

Mit jedem Update-Aufruf des Partikelsystems werden die Parameter der Partikel entsprechend der Einstellungen aktualisiert. Sie erhalten einen veränderten Energiewert, werden weiter bewegt und „altern“. Andauernde Effekte, sowie kurzlebige sind mit den Einstellungsparametern möglich. Zur Visualisierung kommen Point sprites zum Einsatz, diese werden im fünften Kapitel erläutert.

5 Verwendete Shader

Die verwendete Grafikengine jVR macht den Einsatz von Shadern in einer frei konfigurierbaren Render-Pipeline möglich. Die geschriebenen Shader liegen in separaten Dateien dem Projekt bei und wurden in der Shadersprache GLSL verfasst.

5.1 Ball

Abbildung 5.1: Spielball mit Ball-Shader

Der Spielball wird mit dem Shader gerendert, welcher in den Übungen im Modul Computer Graphics and Effects erstellt wurde. Dieser lässt eine golfballähnliche Struktur erkennen, ohne dabei Geometrie zu erzeugen oder besonders viele Vertices zu benötigen. Eine extra Textur ist ebenfalls nicht nötig. Lediglich aus den Texturkoordinaten werden die Vertiefungen berechnet und in Form von veränderten Normalen an den jeweiligen Stellen deutlich.

5.2 Energy bar

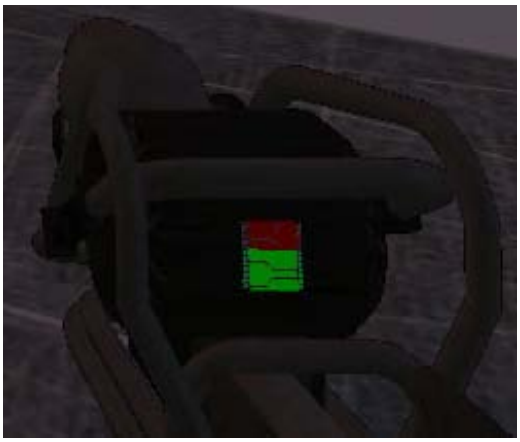


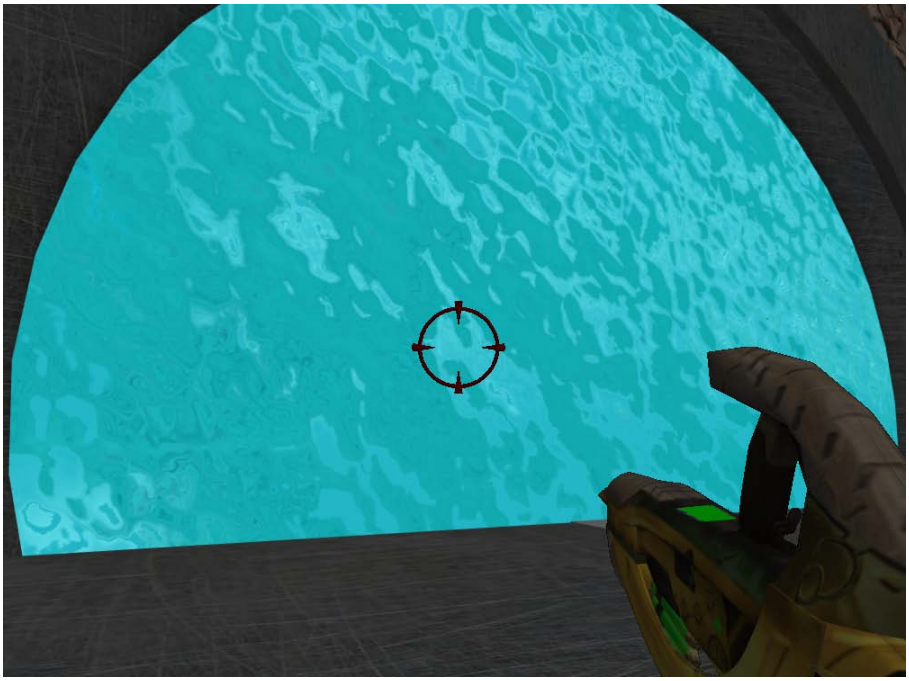
Abbildung 5.2: Energy bar auf den Waffen

Waffen in Powerpong besitzen eine Energieanzeige. Sie zeigt zu jeder Zeit den allgemeinen Energiestand des Spielers. Hierfür wurde eine Ebene an die jeweilige Stelle auf dem Waffenmodell platziert und mit einem eigenen Energy bar-Shader gerendert. Dieser verwendet eine Textur für die Konturen und einen ständig aktualisierten float-Wert um den Stand der Energie zu visualisieren.

Die verwendeten Modelle für die Waffen sind im Vergleich zum Spielermodell recht groß, damit sie in der Egoperspektive gut zu sehen sind. Aus diesem Umstand heraus würden die Waffen den Boden durchstoßen und teilweise verdeckt sein, wenn sich der Spieler den Boden anguckt. Dieser ungewollte Effekt wird durch einen gesonderten Renderdurchgang in der Pipeline umgangen.

5.3 Portal

Die beiden Portale unterscheiden sich optisch aber nicht spielerisch. Das Portal unter der Anzeigetafel gewährt einen Blick in die parallele Welt. Diese ist so realisiert, dass auf eine Plane eine 2. Kameraperspektive gerendert wird. Diese entspricht fast genau der Spieler-Perspektive, ist allerdings um eine Spielfeldbreite auf der X Achse verschoben. Dadurch entsteht der Eindruck, dass der Spieler durch das erste und aus dem 2. Portal herausguckt.



Das gegenüberliegende Portal ist nicht durchsichtig. Ein Parallax-Shader, dessen Normal-Map und Height-Map zyklisch auf der Y Achse verschoben werden, lässt den Stargate-ähnlichen Wasser-Effekt entstehen. Geht der Spieler näher an dieses Portal heran, so kann er den Schriftzug Powerpong und ein statisches Bild des Spielfeldes sehen.

5.4 Point Sprites

Bei Powerpong werden die einzelnen Partikel eines Partikelsystems mit Point Sprites dargestellt. Auf der CPU werden lediglich die Positionen von einzelnen Punkten angegeben. Ein Geometry Shader erzeugt an den Positionen der Punkte später beim Rendern Vierecke. Diese Vierecke werden mittels einer konfigurierbaren Textur gerendert. Der Alpha Wert wird anhand der Energie des jeweiligen Partikels angepasst.

5.5 Environment Mapping

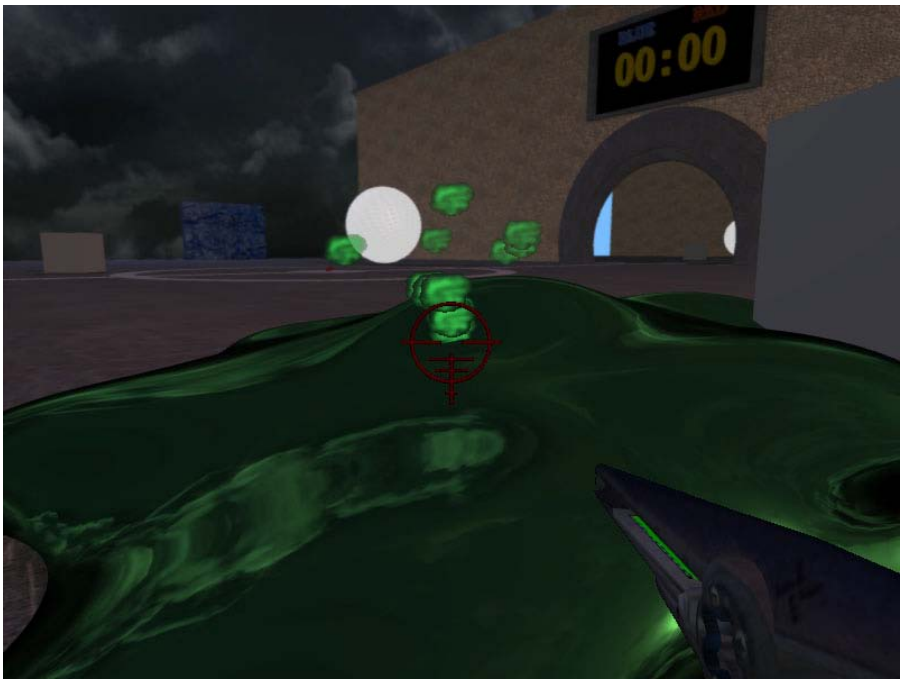


Abbildung 5.3: Environment Mapping bei Powerpong (Gluepaste)

Die Gluepaste wird mit einem Environment-Mapping-Shader gerendert. Dabei wird eine spiegelnde Oberfläche simuliert. Hierfür werden Reflektionsstrahlen berechnet (siehe folgende Abbildung) und an der entsprechenden Stelle in der Skybox nachgeschaut um den jeweiligen Farbwert zu erhalten. Der Charakter der Gluepaste wird verstärkt, indem der Grünanteil verstärkt berücksichtigt wird.

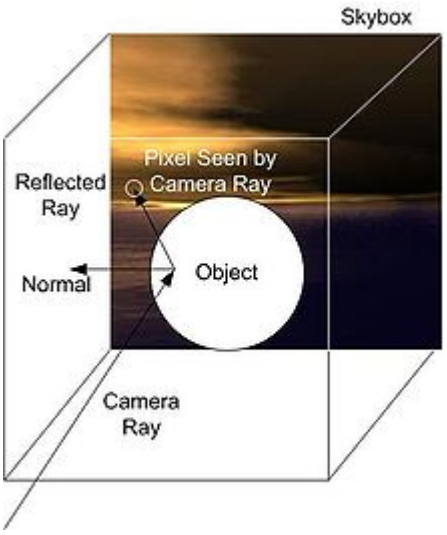


Abbildung 5.4: Farbwertermittlung beim Environment Mapping.
Quelle: http://de.wikipedia.org/wiki/Environment_Mapping

5.6 Parallax Mapping

Parallax Mapping ist eine Erweiterung des Normal Mapping. Beim Normal Mapping wird eine weitere Textur neben den Farbinformationen benötigt. In dieser werden die Normalen Informationen an den jeweiligen stellen kodiert. Das Verfahren ermöglicht es somit, ein erhöhtes Detailreichtum auf den Objekten mit wenigen Vertices darzustellen. Beim Parallax Mapping kommt eine weitere Textur zum Tragen. Diese enthält Informationen für die Höhenunterschiede. Die Geometrie selber wird nicht verändert, lediglich die Farbberechnungen simulieren ein anderes Verhalten, als das einer glatten Oberfläche. Powerpong nutzt diesen Effekt bei den Spielfeldwänden.

5.7 Shadow Mapping

Das Shadow Mapping Verfahren wird in Echtzeit bei Powerpong berechnet. Es ermöglicht einen realistisch wirkenden Schattenwurf aller Objekte in der Szene. Ausgehend von den Lichtquellen wird ein Buffer Objekt berechnet, die Shadow Map. In ihr wird die Sichtbarkeit zu einer Lichtquelle kodiert. Sind Stellen nicht sichtbar, so liegen sie im Schatten. Diese Information wird dafür genutzt, die jeweiligen Farbanteile zu verstärken / reduzieren.

6 Postprocessing

Verschiedene Effekte werden auf die fertig gerenderte dreidimensionale Szene angewandt. Hierfür werden hauptsächlich die Pixelinformationen verwendet.

6.1 Color inversion

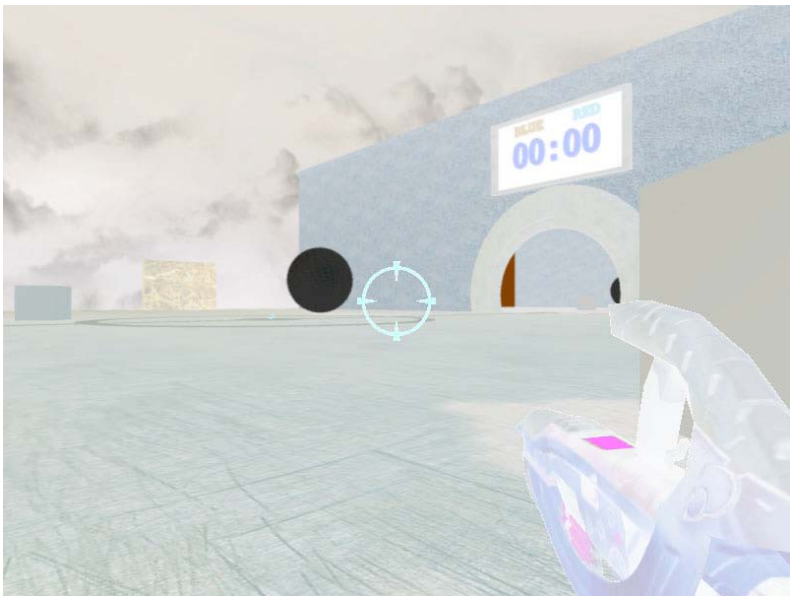


Abbildung 6.1: Color inversion bei Powerpong

Dieser simple Shader invertiert lediglich die einzelnen Farbwerte aller Pixel. Gedacht war dieser Effekt beispielsweise als zufälliges Ereignis. Aus balancing Gründen kann der Benutzer jedoch selber darüber bestimmen. Der Effekt wird mit der Taste „i“ de/aktiviert.

6.2 Bloom

Abbildung 6.2: Bloom bei Powerpong

Dieser Effekt ist auch unter dem Namen Glow bekannt. Helle Stellen im Bild werden dabei noch etwas heller und weicher dargestellt. Der Benutzer kann durch das Drücken der Taste „o“ diesen Effekt ein und ausblenden. Spielerisch hat es keine Auswirkungen.

6.3 Nightvision

Abbildung 6.3: Nightvision bei Powerpong

Mit der Nightvision wird ein Nachtsichtgerät simuliert. Dabei werden zwei Texturen verwendet. Eine um die begrenzte Sicht durch ein Nachtsichtgerät nach zu ahmen. Die Andere wird dazu verwendet, eine gewissen Störung im Bild an zu zeigen. Der Farbwert wird wie im folgenden Listing gezeigt berechnet:

Listing 6.1: Nightvision Shader

```
void main (void)
{
      vec3 color = vec3(1.0, 0.0, 0.0);
      if (withNightVision)
      {
        vec2 uv;
        uv.x = 0.4*sin(elapsedTime*50.0);
        uv.y = 0.4*cos(elapsedTime*50.0);
        float m = texture2D(maskTex, texCoord.st).r;
        vec3 n  = texture2D(noiseTex, (texCoord.st*3.5) + uv).rgb;
        vec3 c  = texture2D(jvr_Texture0, texCoord.st + (n.xy*0.005)).rgb;

        float lum = dot(vec3(0.30, 0.59, 0.11), c);
        if (lum < luminanceThreshold)
          c *= colorAmplification;

        vec3 visionColor = vec3(0.1, 0.95, 0.2);
        color.rgb = (c + (n*0.2)) * visionColor * m;
      }
      else
      {
        color = texture2D(jvr_Texture0, texCoord).rgb;
      }
      gl_FragColor = vec4(color, 1.0);
}
```

Die spielerische Auswirkung ist die Möglichkeit Objekte des Object activators und die Ghosttrain anzuzeigen.

6.4 Pixelation

Abbildung 6.4: Pixelation bei Powerpong

Bei diesem Postprocessing-Effekt werden die angezeigten Pixel in einer wesentlich gröberen Struktur angezeigt. Der Farbwert von beispielsweise 10 mal 10 großen Feldern ist dabei identisch. Eine Uniform-Variable ermöglicht das Einstellen der Größe dieser Felder. Somit kann das Bild beispielsweise über eine gewisse Zeit hinweg immer feiner angezeigt werden, bis es die tatsächliche Auflösung erreicht. Verlässt der Spieler das Spielfeld oder betritt ein Spiel zum ersten Mal, wird dieser Effekt eingesetzt.

6.5 Depth of Field

Depth of Field ist ein Postprocessing-Effekt, welcher zusätzlich zu den Farbinformationen die Tiefeninformationen einer gerenderten Bildes benötigt. Bildbereiche, welche vom Spieler weiter entfernt in der dreidimensionalen Szene existieren, werden hierbei durch eine Art des weich Zeichnens „verwaschen“. Diese Unschärfe simuliert die Unfähigkeit des menschlichen Auges den gesamten Blickbereich zu fokussieren.

7 Installationsanleitung

Das Projekt wird in zwei Versionen ausgeliefert. Zum Einem das Eclipse Projekt, bei dem alle Quelldateien vorhanden sind. Das Projekt ist hier von der Renderengine jVR abhängig. Die Konfiguration muss wie im Wiki von jVR beschrieben vorgenommen werden.

Eine weitere Möglichkeit Powerpong zu starten, ist über die mitgelieferte ausführbare Jar-Version. Hier werden für das starten unter Windows zwei Bat-Dateien zur Verfügung gestellt (eine zum Starten des Servers und einem für den Client). Bei dieser Version sind keine weiteren Einstellungen nötig. Wichtig ist lediglich das Starten mit einer 32-Bit Java VM.

Abbildungsverzeichnis

Abbildung 3.1: Spiele Klassiker Pong 3
Abbildung 3.2: Darstellung des umgesetzten Prorotyps zu Powerpong 4
Abbildung 4.1: Die Peagun 7
Abbildung 4.2: Die Gluegun 7
Abbildung 4.3: Der Object activator 8
Abbildung 5.1: Spielball mit Ball-Shader 11
Abbildung 5.2: Energy bar auf den Waffen 12
Abbildung 5.3: Environment Mapping bei Powerpong (Gluepaste) 14
Abbildung 5.4: Farbwertermittlung beim Environment Mapping. Quelle: http://de.wikipedia.org/wiki/Environment_Mapping 14
Abbildung 6.1: Color inversion bei Powerpong 16
Abbildung 6.2: Bloom bei Powerpong 17
Abbildung 6.3: Nightvision bei Powerpong 17
Abbildung 6.4: Pixelation bei Powerpong 18

Tabellenverzeichnis

Tabelle 3-1: Waffenarsenal des Spielers in Powerpong 3

Listingverzeichnis

Listing 4.1: Auszug aus der Methode checkGluePaste in der Klasse PhysicManager 8
Listing 4.2: Der Konstruktor des Partikelsystems 10
Listing 6.1: Nightvision Shader 18