1 Vorwort

Partikelsysteme sind heute in jedem Computerspiel zu finden. Sie stecken hinter Effekten wie Feuer, Explosionen, Blitzen und Blutspritzern. Aber auch Wasser oder Stoffbewegungen können mit ihnen sehr gut simuliert werden.
Erstmals vorgestellt wurden Partikelsysteme 1983 von William T. Reeves zum Einsatz als Spezialeffekt im Film Star Trek 2 - Der Zorn des Kahn. Ihr großer Vorteil ist, dass sie in ihrer Grundform schon zur Simulation verschiedenartiger Effekte benutzt werden können, ohne dass das ganze System umgeschrieben werden muss. Ein Nachteil dieser Systeme ist der hohe Speicheraufwand, der bei einer sehr hohen Anzahl von Partikeln schnell überhand nehmen kann.

Ziel dieser Arbeit ist die Entwicklung eines Systems für Effekte wie Explosionen, Feuer oder Rauch. Das Demoprogramm konzentriert sich auf die Simulation einer Explosion. Es lässt sich aber auch relativ leicht für andere Anwendungen verändern.

1 Vorwort 1

2 Grundlagen eines Partikelsystems 3

2.1 Attribute eines Partikels 3
2.2 Lebenszyklus eines Partikels 4

2.2.1 Geburt 4
2.2.2 Leben 4

2.3 Tod 5
2.4 Rendern der Partikel 5
2.5 Samplingmethoden 5
2.6 Programmiersprache 6

3 Datenstruktur und Funktionsweise des Programms 7

3.1 Datenstruktur des entwickelten Partikelsystems 7
3.2 Die Klassen 9

3.2.1 Die Partikel-Klasse 9
3.2.2 Partikelsystem-Klasse 10
3.2.3 Die Partikelmanager-Klasse 12
3.2.4 OpenGL-Klasse 13
3.2.5 GUI 14
3.2.6 Hilfsklassen 14

3.3 Ablauf des Programms 15

4 Ergebnisse 16

4.1 Partikelemitter 16
4.2 Simulation einer Explosion 18
4.3 Simulation eines Feuerwerks 20
4.4 Simulation von Feuer 22
5 Fazit 23

6 Quellenangaben 24

2 Grundlagen eines Partikelsystems

Ein Partikelsystem besteht aus vielen einzelnen Partikeln, die ein nicht genau bestimmtes Objekt simulieren sollen. Diese Objekte sind nicht statisch, sondern verändern sich ständig. Deshalb werden in jedem Frame neue Partikel in das System geboren, schon vorhandene Partikel werden modifiziert, und Partikel die ihr Höchstalter erreicht haben sterben.

2.1 Attribute eines Partikels

Jedes Partikel verfügt über Attribute, um sein Verhalten im System zu kontrollieren:

- Farbe
- Form
- Größe
- Position
- Alte Position
- Geschwindigkeit
- Alter

Farbe, Form und Größe bestimmen das Aussehen des Partikels auf dem Bildschirm, und damit auch das Aussehen des ganzen Objektes.
Die Position des Partikels wird in Weltkoordinaten angegeben. Die alte Position eines Partikels kann zur Simulation von Bewegungsunschärfe (Motion Blur) benutzt werden, indem man die alte Position und die neue Position verbindet.
Seine Geschwindigkeit gibt die Richtung und den Betrag an, in die sich das Partikel in jedem Frame bewegt.
Mit dem Alter wird die Lebenszeit des Partikels angegeben.

2.2 Lebenszyklus eines Partikels

2.2.1 Geburt

In jedem Frame werden im System neue Partikel generiert. Die Anzahl der Partikel bestimmt wie dicht das System wird. Dabei wird aber die maximale Anzahl der Partikel im System nie überschritten.

Bei seiner Geburt werden den Attributen jedes Partikels Startwerte zugeordnet. Die von William Reeves verwendete Methode arbeitet mit stochastischen Werten. Für jedes Attribut werden Mittelwert und Standardabweichung festgelegt, die noch mit einem Zufallswert modifiziert werden.

$$Startwert = Mittelwert + Rand(\ )*Variation$$

Je nach verwendetem System kann ein Partikel auch wieder aus einem kompletten Partikelsystem bestehen. Dessen Partikel werden dann mit Startwerten initialisiert, die sich an den Werten des Muttersystems orientieren, sich aber von diesen Werten unterscheiden.

2.2.2 Leben

In jedem Frame werden die Partikel die am Leben sind modifiziert. Jedes Attribut kann sich dabei von einem Frame zum nächsten verändern.
Die Position wird aktualisiert, indem der Geschwindigkeitsvektor zur aktuellen Position addiert wird. Dabei werden auch noch andere Faktoren mit einberechnet, um das System realistischer erscheinen zu lassen. Dazu gehört zum Beispiel die Gravitation, aber auch jede andere Art von Beschleunigung oder Verzögerung. Kollisionsabfragen mit anderen Partikeln oder Objekten können an dieser Stelle auch durchgeführt werden.
Das Alter jedes Partikels wird erhöht, daher nimmt die Restenergie ab.
Farbe und Größe werden verändert, wenn man zum Beispiel das Abkühlen eines Partikels bei einer Explosion simulieren will.

2.3 Tod

Wenn ein Partikel zu alt wird, wird es aus dem System entfernt. Das geschieht, wenn die Restenergie eines Partikels kleiner gleich null wird. Partikel können aber auch schon früher aus dem System entfernt werden, wenn sie nichts mehr zu dem System beitragen. Das ist der Fall, wenn die Kombination aus Partikelfarbe und angewendetem Blending-Effekt nicht mehr vom Hintergrund zu unterscheiden ist.
Tote Partikel können sofort wiedergeboren werden, oder aber komplett aus dem System genommen werden.

2.4 Rendern der Partikel

Nachdem die Partikel aktualisiert wurden, werden sie auf den Bildschirm gezeichnet. Jedes Partikel hat dabei die einfache geometrische Form, zum Beispiel ein Quadrat, oder ein Kreis. Ein Partikel kann aber auch als Punktlichtquelle dargestellt werden.

2.5 Samplingmethoden

Zur Bestimmung der Geschwindigkeit eines Partikels werden stetig gleichverteilte Zufallswerte benötigt. Dafür wird eine nach der inversen CDF Methode generierte Dichtefunktion verwendet. Durch diese Art der Verteilung werden geometrische Muster ausgeschlossen, die sonst vor allem bei diskreten Verteilungen stark ins Auge fallen würden. Für eine Kreisförmige Verteilung wird die Formel:

$$\frac{1}{\pi * R^2}$$

verwendet. Für andere Verteilungen werden andere Funktionen gebraucht.

2.6 Programmiersprache

Als Programmiersprache wird C++ benutzt, weil es bei einem Partikelsystem durch die vielen Operationen auf Geschwindigkeit ankommt. Zum rendern der Partikel wird OpenGL benutzt. Die Graphische Oberfläche wird mit der Bibliothek QT erzeugt.

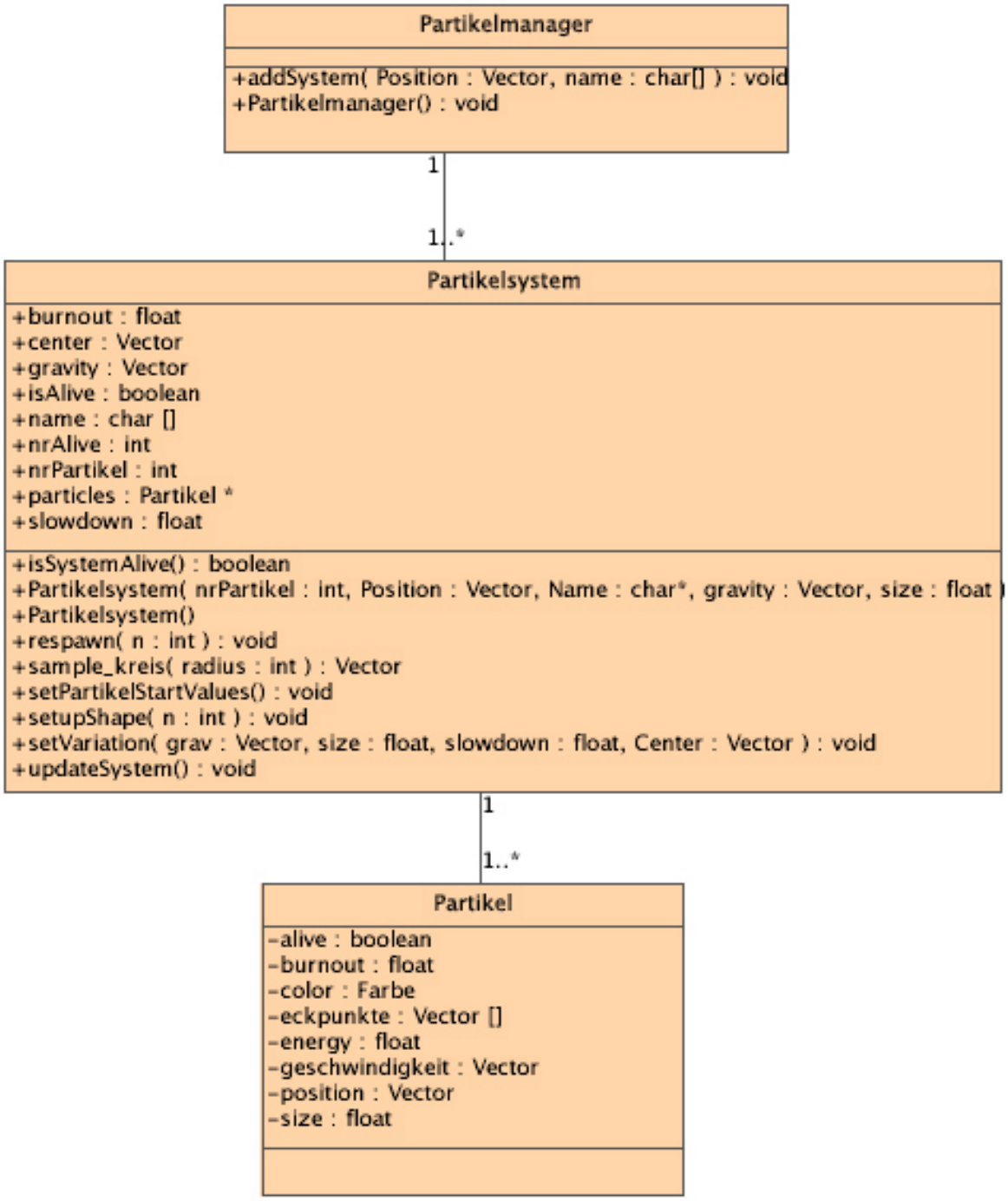
3 Datenstruktur und Funktionsweise des Programms

3.1 Datenstruktur des entwickelten Partikelsystems

Die Datenstruktur des Partikelsystems besteht aus 3 Klassen:

- dem Partikelmanager
- dem Partikelsystem und den
- Partikeln.

Der Partikelmanager kontrolliert die einzelnen Partikelsysteme, die dann die Partikel kontrollieren. Diese Datenstruktur ermöglicht es in einem Programm mehrere Partikelsysteme mit unterschiedlichen Funktionen von einem Punkt aus zu steuern. Die Ausprägungen der einzelnen Systeme, zum Beispiel Blitze oder Explosionen, werden alles von einem Partikelmanager aus gesteuert.


Partikelmanager
+addSystem(Position : Vector, name : char[]) : void
+Partikelmanager() : void
1
1..*
Partikelsystem
+burnout : float
+center : Vector
+gravity : Vector
+isAlive : boolean
+name : char []
+nrAlive : int
+nrPartikel : int
+particles : Partikel *
+slowdown : float
+isSystemAlive() : boolean
+Partikelsystem(nrPartikel : int, Position : Vector, Name : char*, gravity : Vector, size : float)
+Partikelsystem()
+respawn(n : int) : void
+sample_kreis(radius : int) : Vector
+setPartikelStartValues() : void
+setupShape(n : int) : void
+setVariation(grav : Vector, size : float, slowdown : float, Center : Vector) : void
+updateSystem() : void
1
1..*
Partikel
-alive : boolean
-burnout : float
-color : Farbe
-eckpunkte : Vector []
-energy : float
-geschwindigkeit : Vector
-position : Vector
-size : float

3.2 Die Klassen

3.2.1 Die Partikel-Klasse

Die Partikelklasse enthält die schon beschriebenen Attribute eines Partikels:

Typ	*Name*	*Beschreibung*
Vector	position	Position des Partikels in Weltkoordinaten
Vector	velocity	Geschwindigkeitsvektor des Partikels
bool	isalive	Lebt das Partikel
float	size	Größe des Partikels
float	energy	Energie des Partikels
float	burnout	Geschwindigkeit mit der das Partikel altert
Farbe	color	Farbe des Partikels
Vector[]	vertex	Eckpunkte des Partikels

Die Energie des Partikels entspricht dem Alter des Partikels, das heißt, wie lange es noch zu leben hat. Bei einer Explosion wird simuliert, wie die einzelnen Partikel mit der Zeit verglühen. Dazu wird eine Blending Funktion zwischen der Partikelfarbe und der Hintergrundfarbe benutzt. Die Partikelfarbe wird als RGB Wert mit zusätzlichem Alpha Wert für die Transparenz angegeben. Der Wert 1 beschreibt ein undurchlässiges Partikel, der Wert 0 ein transparentes. Die Funktion blendet die Partikel in den Hintergrund.
Deshalb liegen die Energiewerte der Partikel zwischen 1.0 und 0.0 und nehmen mit zunehmendem Alter ab.
Burnout ist der Wert, der in jedem Frame vom Alter des Partikels abgezogen wird. Er wird zufällig zwischen 0 und 0.004 initialisiert.
Der Vector Array vertex speichert die Eckpunkte der Partikel.

Die Partikel-Klasse enthält keine Funktionen, da sie nur als Datenspeicher für die Partikelattribute dient. Die Funktionen liegen in der Partikelsystem-Klasse.

3.2.2 Partikelsystem-Klasse

Die Partikelsystem-Klasse ist das Herz des Systems, mit den Funktionen zum Initialisieren des Systems, zum Setzen der Eckpunkte und zum Aktualisieren des Systems. Für verschiedene Systeme werden verschiedene Funktionen gebraucht. Die hier vorgestellte Klasse simuliert Explosionen aus quadratischen Partikeln. Die Grundstruktur kann aber auch für jede andere Art von System genutzt werden.

Attribute der Klasse

Die Attribute dieser Klasse beeinflussen das System als Ganzes:

Typ	*Name*	*Beschreibung*
int	nrPartikel	Anzahl der Partikel im System
int	nrAlive	Anzahl der noch lebenden Partikel
char[]	name	Name des Systems
Partikel*	particles	Array der Partikel im System
Vector	center	Ursprung des Systems
Vector	gravity	Zieht die Partikel nach unten
Vector	slowdown	Vector mit dem die Partikel abgebremst werden

Der Wert *nrPartikel* gibt die maximale Anzahl der Partikel im System an, *nrAlive* die Anzahl der noch lebenden Partikel. Wenn *nrAlive* 0 wird, kann das System aus dem Programm entfernt werden, oder komplett neu initialisiert werden. Dadurch werden teure Speicheroperationen gespart.

Die Funktionen

Der Konstruktor initialisiert die wichtigsten Attribute des Systems:

- Die Anzahl der Partikel im System, damit wird der Array initialisiert.
- Der Name des Systems, um die einzelnen Systeme voneinander zu unterscheiden.
- Der Ursprung des Systems, von dem aus die Partikel bei einer Explosion alle starten.

Zum Startzeitpunkt des Systems werden zwei Funktionen gebraucht:

setVariation ist eine Hilfsfunktion zum setzten von Variablen, die das System als Ganzes beeinflussen, und die auch noch zur Laufzeit beeinflusst werden können. Dazu gehören der Ursprung des Systems, der Gravitationsvektor, slowdown, Farbe und Größe der Partikel. Durch diese Funktion kann die Funktion *setPartikelStartValues* ohne Parameter aufgerufen werden. Sie wird für einen komplette Neustart des Systems aufgerufen..

Die Funktion setParticleStartValues setzt für alle Partikel gleichzeitig die Startwerte aller Attribute.

- Die Position wird bei einer Explosion für alle Partikel mit dem Ursprung des ganzen Systems gesetzt. Sie könnte aber auch durch eine stochastische Formel innerhalb einer Region (Generation Shape) berechnet werden.
- Die Geschwindigkeit wird mit der schon beschriebenen Dichtefunktion berechnet. Beide Simulationen, die Explosion und der Emitter werden mit derselben Kreisförmigen Funktion berechnet.
- *isAlive* wird auf true gesetzt
- *burnout* wird wie schon erwähnt zufällig zwischen 0 und 0.04 berechnet
- *energy* wird auf 1.0 gesetzt.
- Die Variablen *color*, *size* , *slowdown* und *gravity* werden mit den Standardwerten des Systems initialisiert.

Die Funktion *setupShape* berechnet die Eckpunkte der Partikel. Jedes Partikel wird quadratisch um die aktuelle Position herum berechnet. Der Durchmesser des Partikels entspricht dabei seiner Größe. In jede Richtung wird die halbe Größe dazu addiert oder abgezogen.

Da bei diesem System ohne Kamera gearbeitet wird, ist es nicht nötig die View-Matrix mit einzubeziehen.

Die Aktualisierungsfunktion *updateSystem* wird in jedem Frame vom Hauptprogramm aus aufgerufen. Hier wird für jedes Partikel die neue Position berechnet:

$$position = altePosition + velocity$$

Das Alter des Partikels wird über den Wert *burnout* vermindert. Fällt die Restenergie dabei unter Null, wird *isAlive* auf *false* gesetzt.
Bei der Simulation einer Explosion werden hier auch noch Geschwindigkeit und Größe der Partikel verändert. Beide Werte werden mit der Zeit verringert.
Wird ein Emitter simuliert werden die Partikel an dieser Stelle wiedergeboren.

Dies geschieht mit der Funktion *respawn*. Ähnlich wie bei *setParticleStartValues* werden hier die Attribute der Partikel wieder auf ihre Startwerte zurückgesetzt, nur das *respawn* nur auf einem Partikel arbeitet, und nicht für alle Partikel gleichzeitig arbeitet.

3.2.3 Die Partikelmanager-Klasse

Diese Klasse kontrolliert die einzelnen Partikelsysteme. Sie enthält keine Attribute, sondern nur Funktionen.

Der Konstruktor dieser Klasse ist leer.

Die Funktion *addsystem* fügt ein Partikelsystem mit angegebenem Namen, an der angegebenen Position hinzu. Dazu wird erst ein neues Partikelsystem angelegt. Dann werden die Funktionen *setVariation* und *setParticleStartValues* aufgerufen.

Die Funktion *deletesystem*, die zum löschen eines toten Partikelsystems dient, wird in diesem Programm nicht gebraucht, und ist deshalb nicht ausprogrammiert.

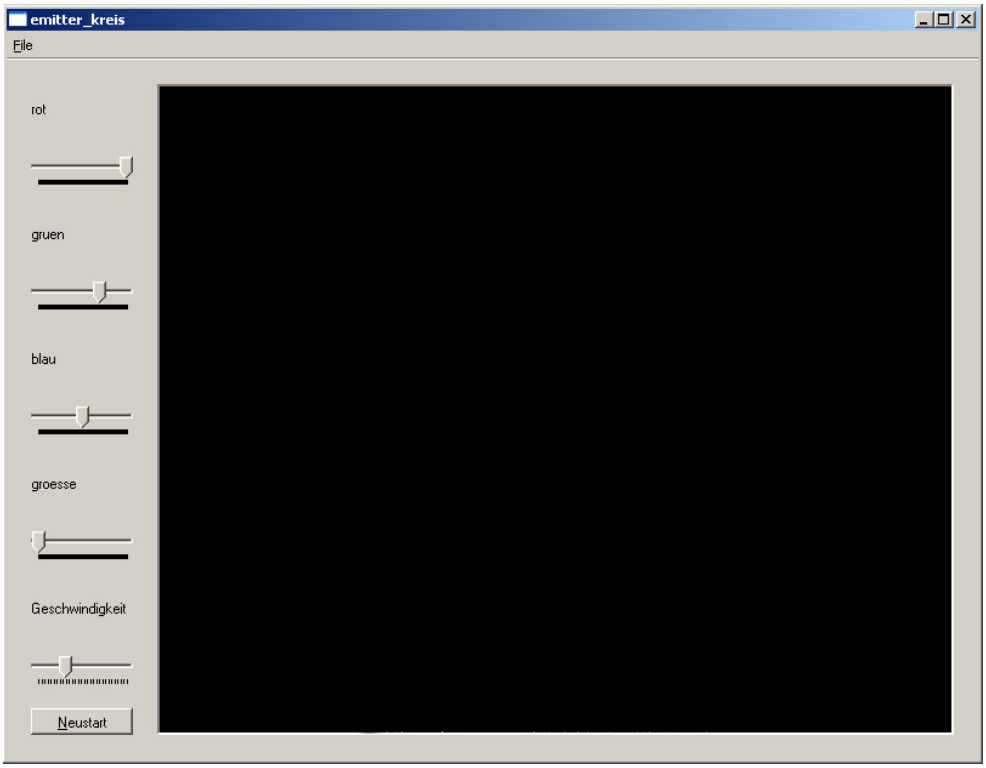
3.2.4 OpenGL-Klasse

Die Klasse glfenster enthält Funktionen zum Zeichnen der Partikel auf dem Bildschirm, und zur Anbindung der GUI an die Partikelsysteme.

Zeichnen der Partikel

Für jedes Partikel im System wird zuerst überprüft, ob das Partikel noch am Leben ist. Wenn das der Fall ist, werden die Eckpunkte des Partikels gesetzt. Die Zeichenfarbe des Partikels wird festgelegt, als Blending Wert wird die Restenergie des Partikels benutzt. Die vier Eckpunkte des Partikels werden als GLQuads gezeichnet.
Nachdem auf diese Weise das System komplett gezeichnet worden ist, wird die Aktualisierungsfunktion aufgerufen.
Durch die Anbindung an QT in diesem Programm gibt es noch einen Timer, der die Zeit bis zum erneuten Ausführen der Funktion angibt. Dieser Timer ist Teil der QT-Bibliothek.

3.2.5 GUI



Die GUI des Programms, bestehet aus dem OpenGL Fenster, in dem die Partikel gezeichnet werden, den Slidern zum Verändern von Farbe und Größe der Partikel, und einem Knopf zum Neustart des Systems..
Die Funktionen hierzu werden von der Bibliothek QT geliefert. Die einzelnen Slider sind mit Funktionen in der Klasse *glfenster* verknüpft, und setzten die entsprechenden Variablen des Partikelsystems neu.
Bei einem Neustart des Systems wird die Funktion *setParticlesStartValues* des Partikelsystems aufgerufen.

3.2.6 Hilfsklassen

Als zusätzliche Datenstrukturen werden noch zwei weitere Klassen benötigt. Die Klasse *Farbe* stellt Funktionen zum Verwalten der Partikelfarbe im RGB Raum zur Verfügung. In der Klasse *Vector* sind Funktionen zum Verwalten von Vektoren untergebracht.

3.3 Ablauf des Programms

Als erstes wird das GUI –Fenster angelegt. Dann wird die Klasse *glfenster* aufgerufen. Dort wird der Partikelmanager angelegt, durch den die einzelnen Systeme erzeugt werden. Danach läuft die Zeichenfunktion, solange bis das Programm beendet wird.

4 Ergebnisse

4.1 Partikelemitter

Am Beispiel eines Partikelemitters werden die Grundlegenden Eigenschaften des Partikelsystems gezeigt. Von einem Startpunkt aus werden Partikel generiert, die, nachdem sie gestorben sind am Ausgangspunkt wiedergeboren werden. Mit Hilfe von vier Slidern können die Partikelfarbe als RGB-Wert und die Partikelgröße verändert werden. Dazu kommt noch ein Knopf zum Neustart des Systems.

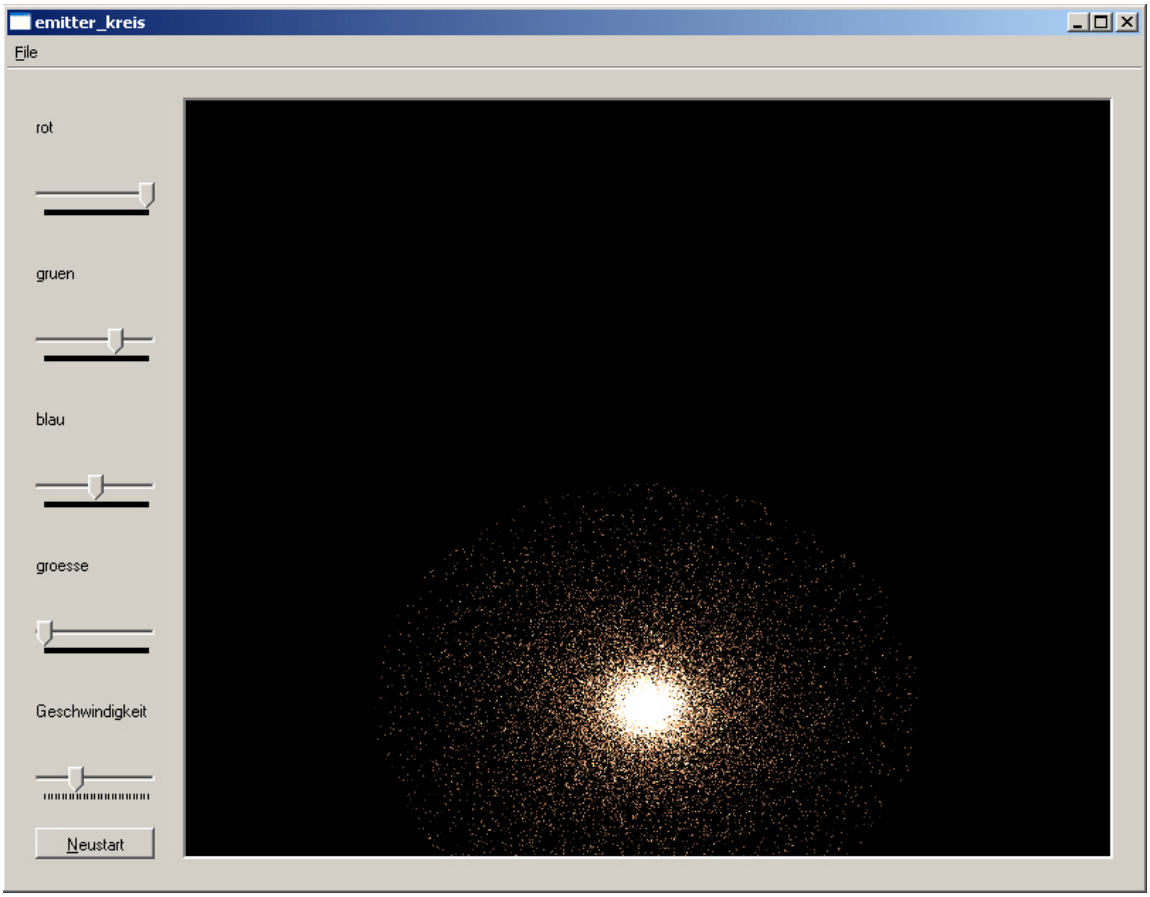
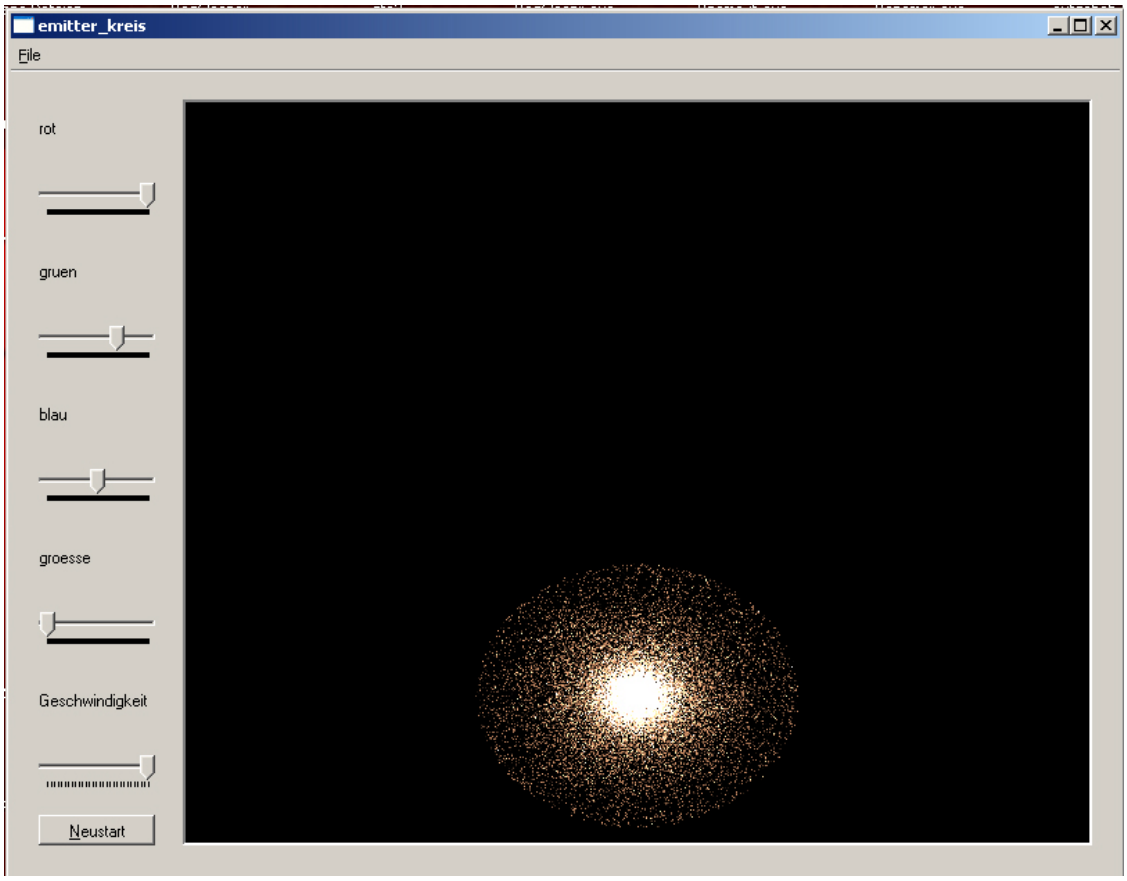
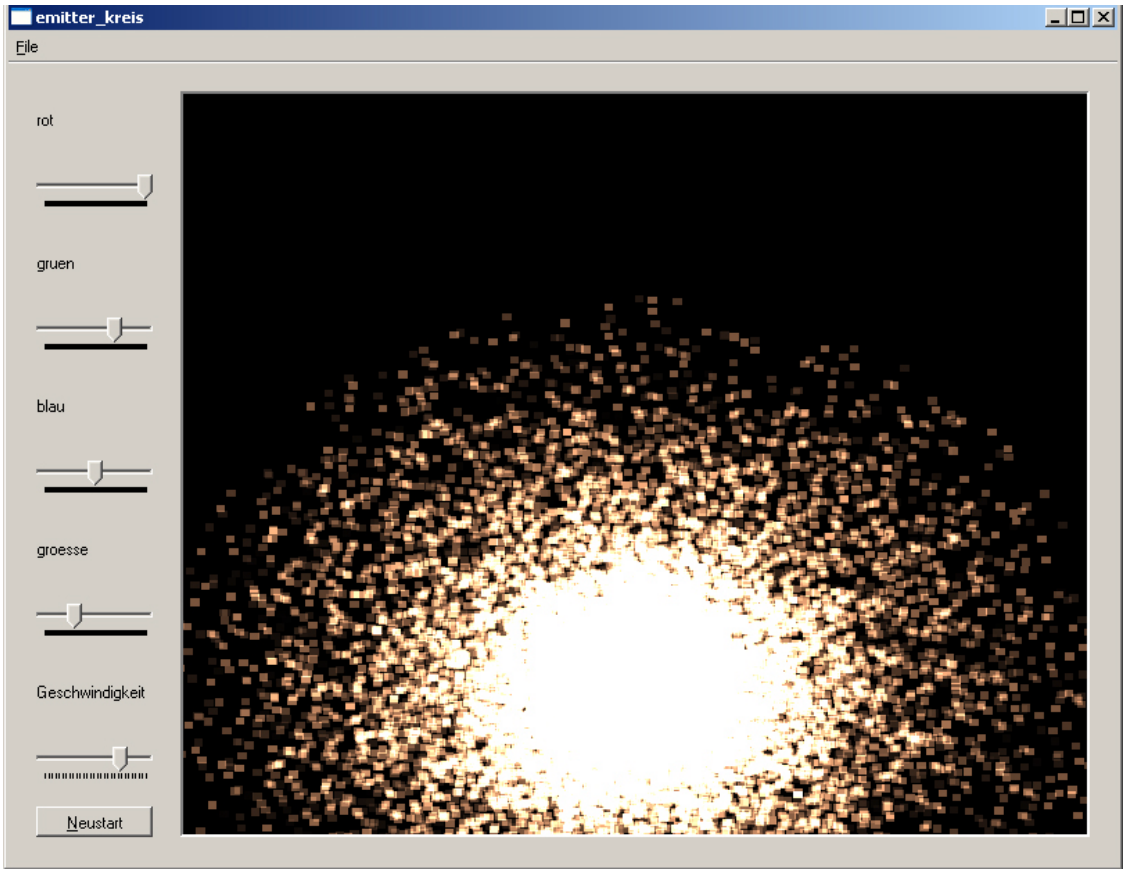


Bild zum Startzeitpunkt eines Emitters



Emitter nach mehreren Durchläufen



Emitter mit größeren Partikeln

4.2 Simulation einer Explosion

Die Partikel werden von einem Punkt aus kreisförmig nach außen beschleunigt. Über die Zeit werden Größe und Geschwindigkeit der Partikel verringert. So wird das Verglühen der Partikel simuliert. Die Partikel werden in dieser Simulation nicht wiedergeboren.

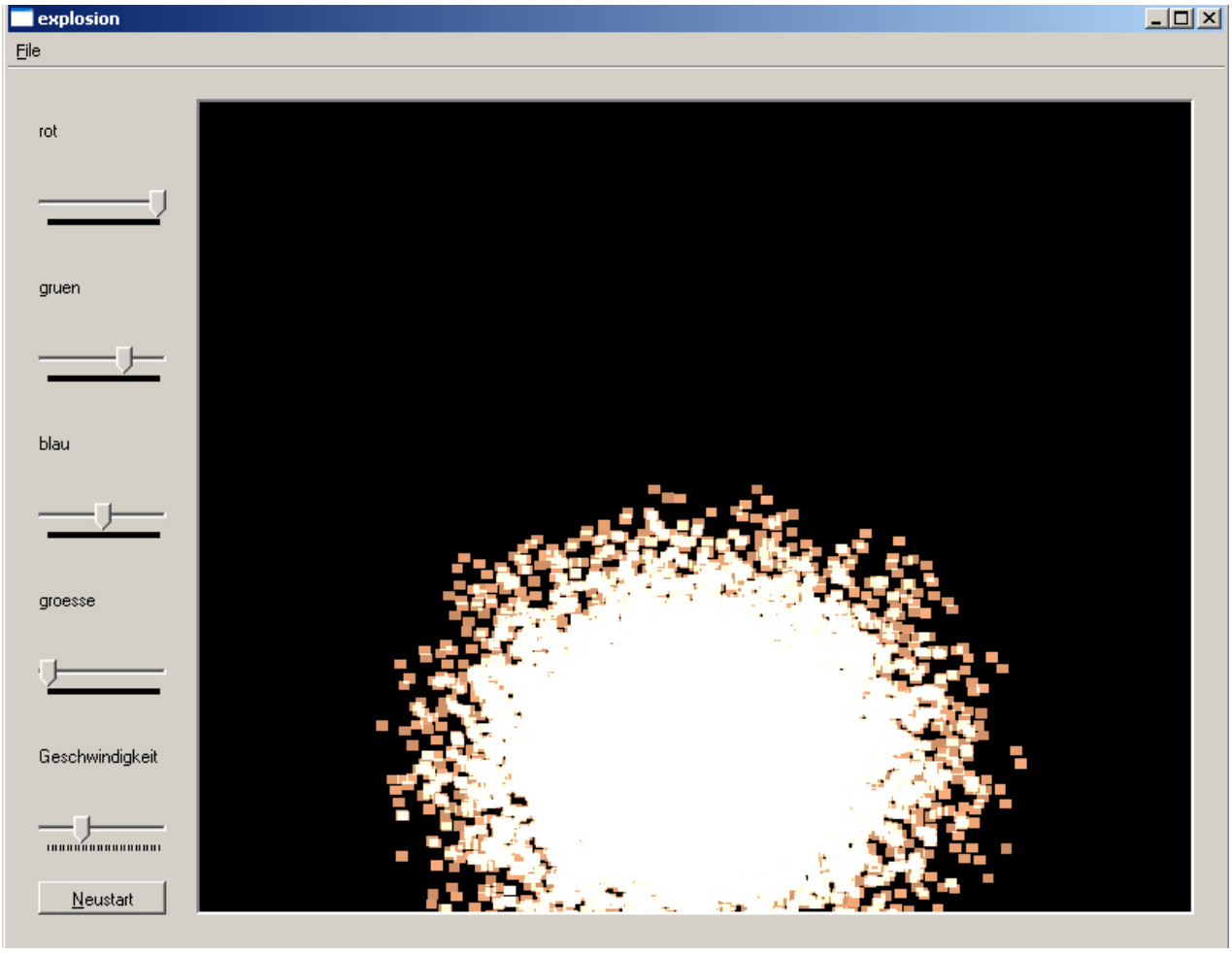
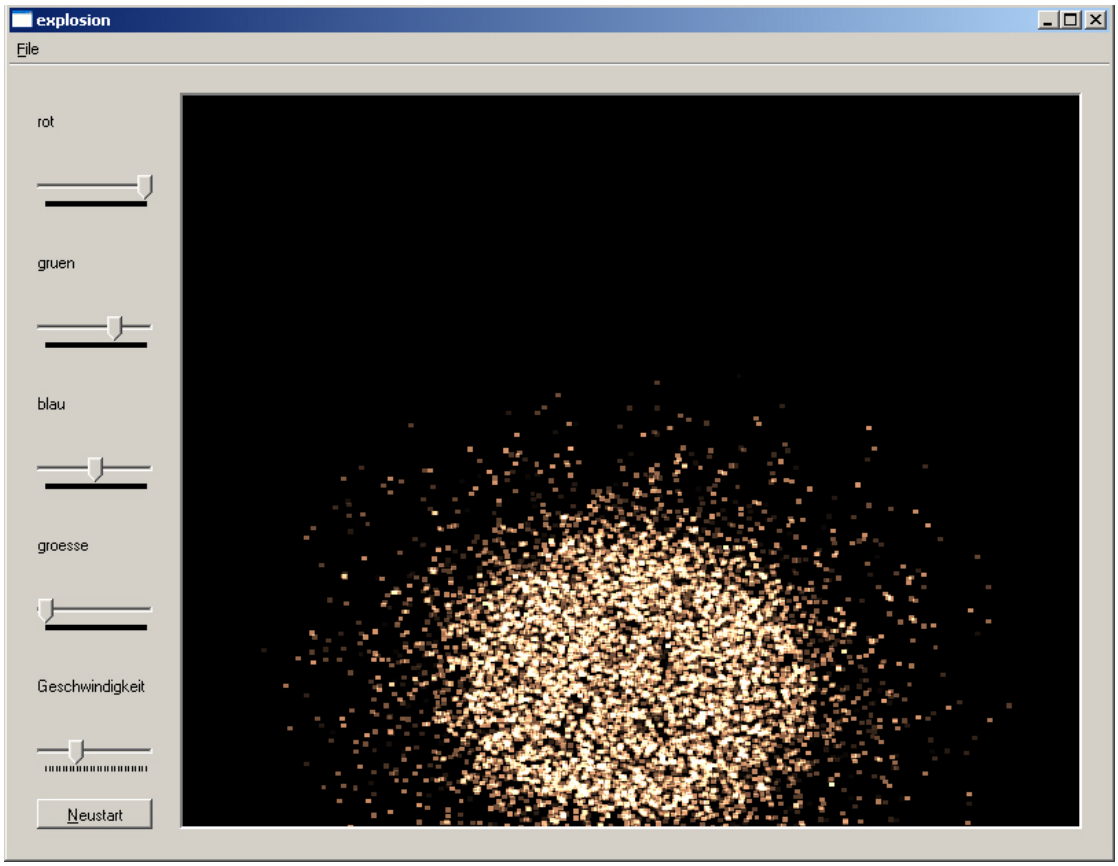


Bild zum Startzeitpunkt der Explosion



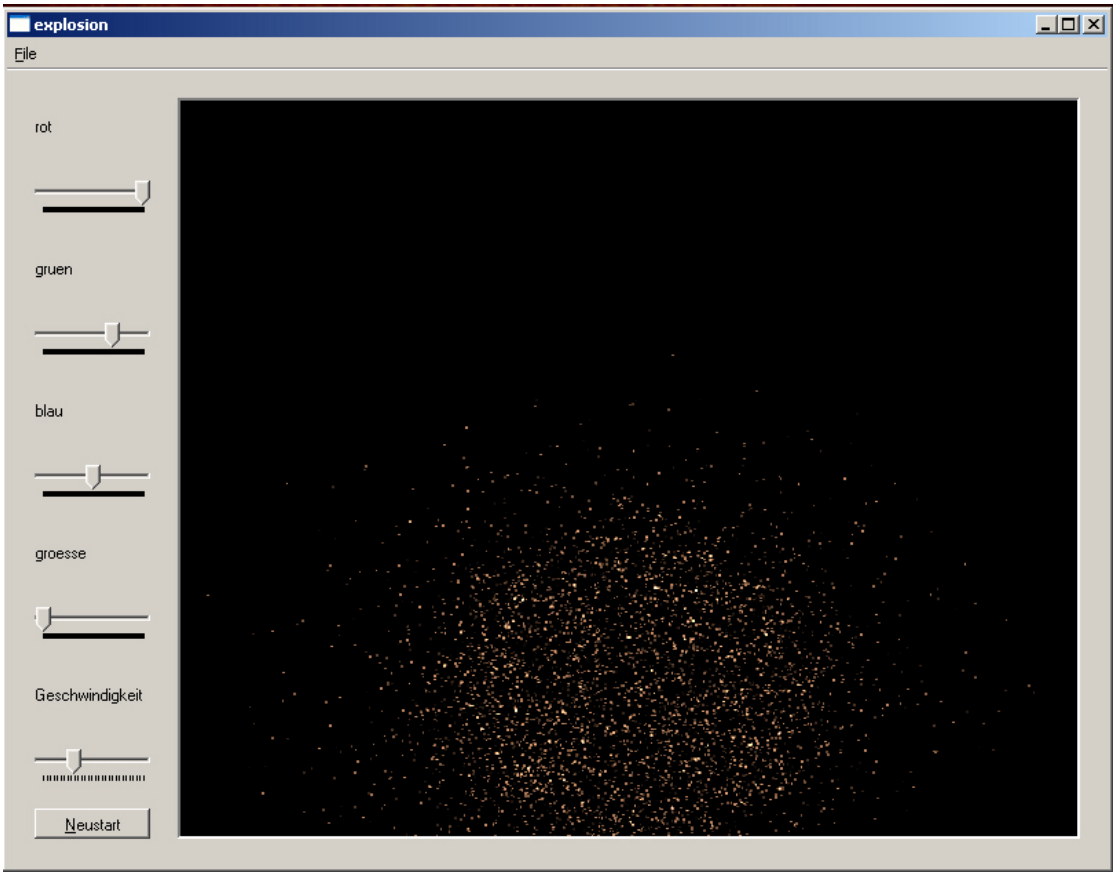
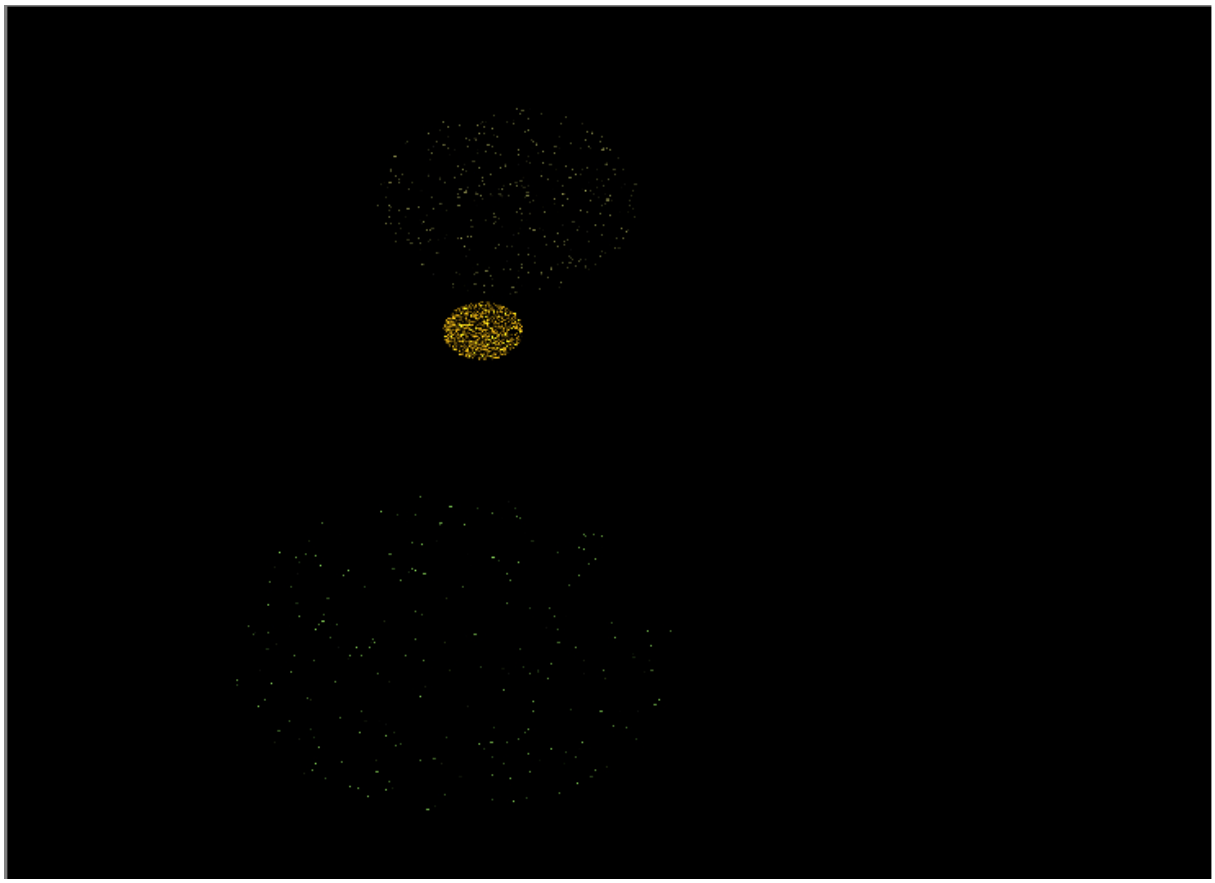


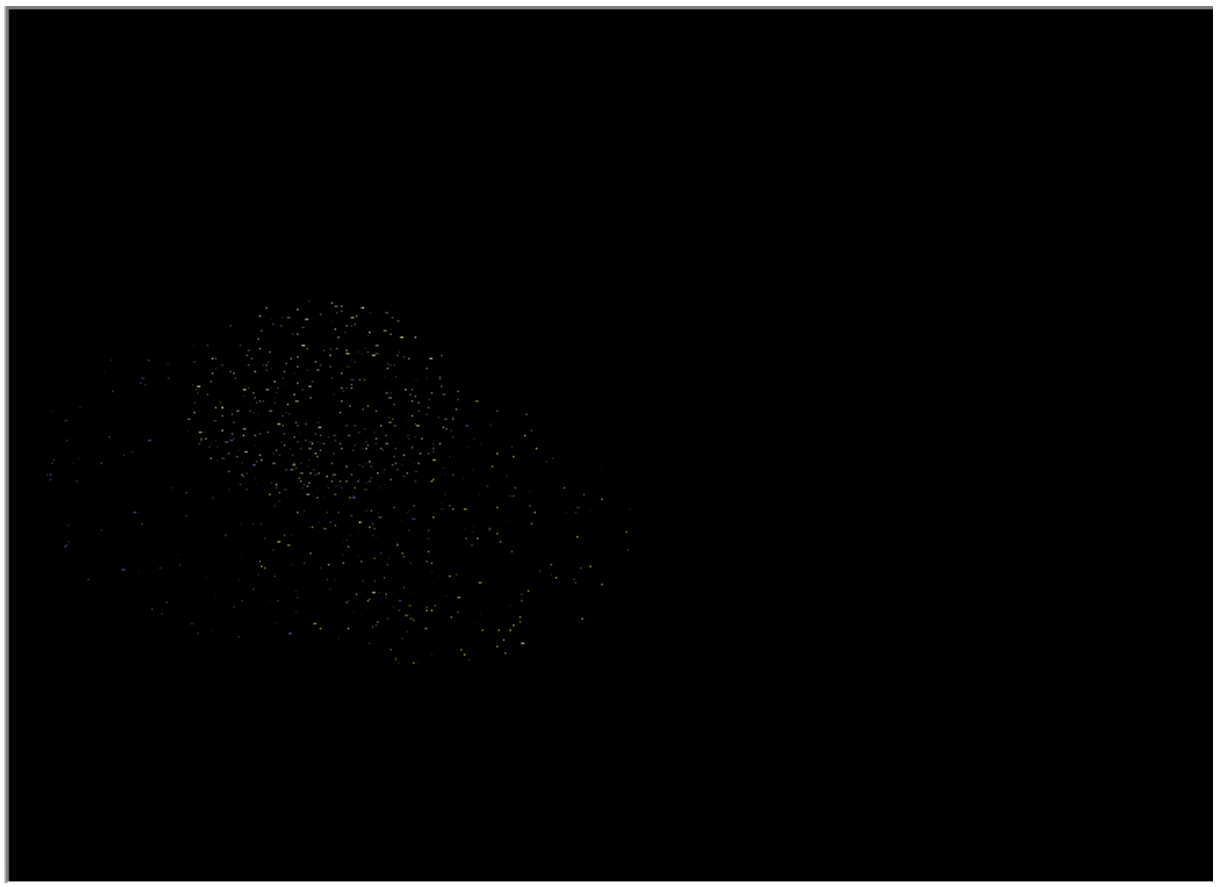
Bild zum Ende der Explosion

4.3 Simulation eines Feuerwerks

In dieser Simulation werden drei Partikelsysteme generiert. In jedem System sind 8.000 Partikel. Die einzelnen Systeme werden nach einer zufällig festgelegten Anzahl von Frames neu gestartet. Dabei werden die Position und die Farbe der Partikel zufällig verändert. Die Gravitation wird mit einberechnet, die Partikel werden nach unten gezogen.



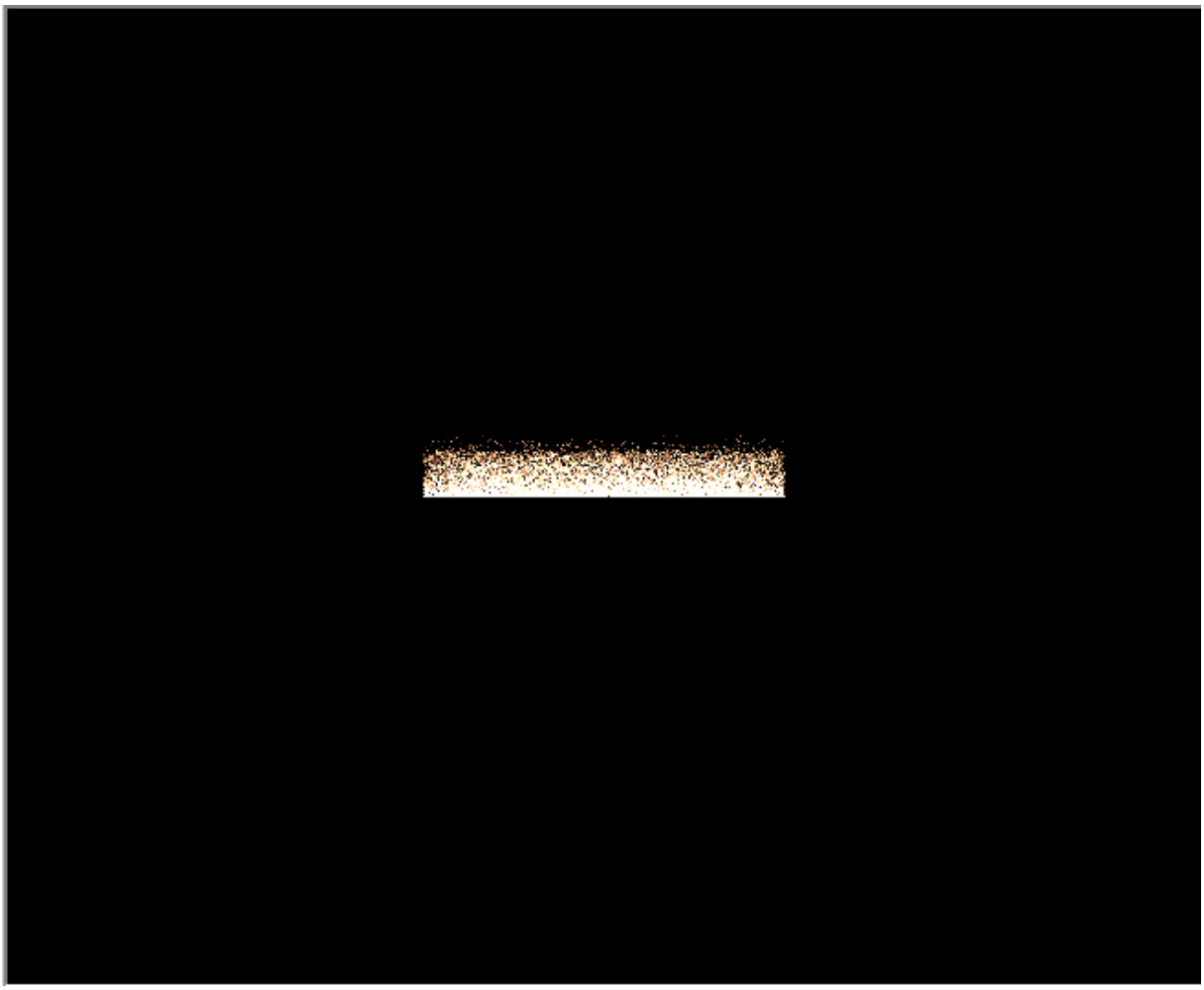
Feuerwerk



Feuerwerk mit Gravitation

4.4 Simulation von Feuer

Feuer wird mit einem System von 100.000 Partikeln simuliert. Die Startpositionen der Partikel liegen auf einer Linie. Von dort werden die Partikel nach oben beschleunigt. Die Partikel werden auf der Grundlinie wiedergeboren. Auch hier wird die Gravitation mit einberechnet. Eine Kollisionsabfrage verhindert, dass die Partikel unter die Grundlinie fallen.



5 Fazit

Das im Rahmen dieser Arbeit programmierte System zeigt die grundlegenden Eigenschaften von Partikelsystemen zur Simulation von Explosionen und Feuer. Die Datenstruktur ist flexibel programmiert, und kann auch in einem größeren Programm eingesetzt werden. Das Partikelsystem ist noch in vielerlei Hinsicht erweiterbar.

- Ein besseres Speichermanagement würde die Laufzeit verbessern.
- Die Partikel können mit einer Textur versehen werden, die den jeweiligen Effekten angepasst ist.
- Andere Samplingfunktionen für Feuer können die Simulation realistischer machen.
- Die Partikelbewegung könnte nach Physikalischen Gesetze vorgehen, anstatt nach stochastischen Werten.

6 Quellenangaben

- Lander, Jeff, The Ocean Spray in your face,
  Graphic Content, July 1998

- Reeves, W.T., Particle systems-A technique for modelling a class of fuzzy objects,
  SIGGRAPH 83, Computer Graphics 17, 3 (July 1984), 359-376

- Reeves, W.T., Approximate and Probabilistic Algorithms for Shading and Rendering
  Strucutured Particle Systems, SIGGRAPH 85, Computer Graphics 85, 3
  (November 1985), 313-322

- Van der Burg, John, Building an advanced Particle System,
  Game Developer Magazine March 2000, 44-50

- Watt, Alan, 3D Computer Graphics.
  Reading, Mass.: Addison Wesley, 1993.