Projektbeschreibung Magneto

Gruppe 3

Mitglieder Christian Blackert (CB)
Claus Liebenberger (CL)
Stefan Reidlinger (SR)

Spielidee

Ziel des Spiels ist es einen Ball mit Hilfe von Magnetfeldern durch ein Level (2d) zu steuern, um ein Ziel zu erreichen.

Der Aufbau des Levels erfolgt über Tiles, welche am Bildschirm gescrollt werden, und einem statischen Hintergrundbild (mit geringerer Scrollgeschwindigkeit für 3D-Effekt).

Im Level gibt es mehrere Magneten, mit deren Hilfe der Ball gesteuert werden kann. Der Spieler kann über einen Tastendruck nur alle Magnete aktivieren oder deaktivieren.

Jedes Magnetfeld (in einem gewissem Umfeld um den Ball) zieht den Ball an und übt eine zusätzliche Kraft in eine der 4 Richtungen (oben/unten/links/rechts) auf den Ball aus. Zusätzlich wirkt in jedem Level grundsätzlich die Schwerkraft.

An bestimmten Positionen im Level sind "gefährliche" Gegenstände (Spitzen), die den Ball zerstören können. Deshalb muss der Spieler darauf achten, dass der Ball diese Gegenstände nicht berührt. Ist dies der Fall, wird der Ball wieder in die Ausgangsposition zurückversetzt, und der Level beginnt von vorne.

Durch Lösen eines Levels gelangt man zum nächsten, welches bis dahin gesperrt war.

Die Komplexität / der Schwierigkeitsgrad steigt mit jedem Level – die Anfangslevel dienen dazu, die Steuerung des Balls zu erlernen.

Die Zeit, die der Spieler benötigt, um ein Level abzuschließen, bestimmt den Score, der Score, bzw. der letzte frei geschaltene Level kann lokal mit einem Benutzernamen gespeichert werden. Dadurch können auch unterschiedliche Spieler mit einer Installation spielen.

Entwicklungsumgebung, Plattform

als Entwicklungsumgebung wird das Microsoft XNA Framework 3.0 mit Programmiersprache C# verwendet.

Zielplattform: Windows XP oder höher mit entsprechenden .NET Framework 3.0.

Gamestates

Das Spiel hat 4 Zustände:

1. Initialisierung: Anzeige eines Splashscreens während des Ladevorganges von Ressourcen. Danach zur Benutzerauswahl.

2. Benutzerauswahl: Bevor ein Spieler mit dem Spiel beginnen kann, muss er seine Initialen bekannt geben. Gibt der Spieler Initialen die das Spiel noch nicht kannte, wird für diesen Spieler ein neues File angelegt (Initialen.dat). Dieses File beinhaltet eine Liste der bereits bewältigten Levels (im Initialfall ist die Datei Empty), sowie der erzielte Score je Level. Diese Informationen sind nötig um den Spieler seine aktuellen Erfolge (Score) anzuzeigen, sowie die Auswahl des Levels zu ermöglichen. Danach weiter zum Menü.
3. Menü: Dieses Fenster zeigt den Score des Spielers an, gibt eine Rückmeldung, ob der letzte Level erfolgreich war oder nicht. Der Spieler kann den nächsten Level auswählen und starten bzw. aus den bereits geschafften und den als nächstes zu bewältigenden Level auswählen.
4. Level: Das eigentlich Spiel endet entweder durch "Aufgabe" (Esc), oder durch Lösen des Levels (Ball erreicht Zielquadrant) -> Nach Beendigung eines Levels wird wieder das Menü angezeigt. Für die Bewältigung des Levels gibt es kein Zeitlimit. Die Score Ermittlung erfolgt dahingehend, dass für jeden Level eine Sollzeit angegeben wird, die der Spieler unterbieten muss um an einen Score zu gelangen. Sollte er über diesem vorgegeben Zeitwert liegen und der Level erfolgreich beendet, wird ein Score von 0 gesetzt.

Beendet wird das Spiel durch Schließen des Fensters (rotes X).

Beschreibung der Komponenten

In den nachfolgenden Abschnitten werden kurz die Funktionalitäten der einzelnen zu implementierenden Klassen beschrieben, um die Schnittstellen zwischen den Modulen, die von unterschiedlichen Personen programmiert werden zu definieren.

Sound

Es gibt 2 unterschiedliche Backgroundsounds (Splash und Menü; Level)

Weiters werden die folgenden Soundeffekte benötigt:

- Klick: für Betätigen eines Steuerelements im Menü
- Magnet: Wenn der Ball von einem Magnet angezogen wird
- Dead: Wenn der Ball mit einem gefährlichen Gegenstand kollidiert
- Success: Wenn der Bass den Zielquadranten erreicht
- Bounce: Wenn der Ball von einer Mauer abprallt.

Über die Taste "M" soll ein Spieler die Hintergrundmusik ein, oder ausschalten können. Das Abspielen der Hintergrundmusik, sowie der einzelnen Sounds wird von der Klasse clsSound übernommen.

Grafik

Der Aufbau eines Levels erfolgt in 3 Layern:

1. Layer: Dieser Layer enthält die Tiles, von welchen der Ball abprallt
2. Layer: Dieser Layer enthält die gefährlichen Gegenstände, die den Ball töten
3. Layer: Dieser Layer enthält die Magnete, die den Ball in eine bestimmte Richtung beschleunigen.

Als Basis für den Bildschirmaufbau wird das Tutorial von http://nickgravelyn.com/archive/#tileengine verwendet.

Das Einlesen eines Levels, sowie dessen Darstellung (Draw-Methode) wird in der Klasse clsCanvas abgearbeitet. Diese Klasse stellt zusätzlich Methoden zur Verfügung, die für einzelne Quadranten, die im Quadranten enthaltenen Vektoren für die Game-Physik zurückliefert.

Der Viewport hat immer eine Größe von 800x600 Pixel. Die Tiles haben eine Größe von 32x32 Pixel. Der Ball hat einen Radius von 10 px.

Jeder Level muß also mindestens 25x18 Tiles enthalten, kann jedoch größer sein. es ist Aufgabe der clsCanvas den Viewport entsprechend der Postion des Balls zu bestimmen. Dabei soll der Ball möglichst in der Mitte des Viewports liegen. Wenn sich der Ball bewegt, soll das Viewport "nachziehen" (d.h. der Ball kann sich – bei höheren Ballgeschwindigkeiten – aus der Mitte des Viewports herausbewegen)

Die Levels werden in Textfiles definiert (ähnlich wie INI-Files) mit den folgenden Abschnitten:

[INIT]
Tiles-X: 50
Tiles-Y: 50

Ball-X: 1

Ball-Y: 2 ' Initiale Ballposition

[BOUNCE]

AAABBCC…..

AAABBCC…..

AAABBCC…..

AAABBCC…..

AAABBCC…..

[MAGNETS] ' X-Pos, Y-Pos, Direction

1,1,LO (1. Tile Magnetkraft richtet sich nach links oben)

[SPIKES]

1,1,U (1.Tile hat einen Spikebereich an der Unterkante des Quadranten)

➔ weitere Möglichkeiten: O, U, L, R, /, \ (/… diagonale Spikes)

Physik

Der Ball unterliegt einer Gravitation, wird (beim Betätigen der Leertaste) von "naheliegenden" Magneten angezogen und zusätzlich in eine bestimmte Richtung "gedrückt".

Die Kollisionsprüfung erfolgt über Schnittpunktbildung von Geradensegmenten. Jedes Geradensegment eines Quadranten (Polygon) besitzt eine Zusatzinformation, ob der Ball abprallen soll, oder ob es sich um eine Spike-Linie handelt (der Ball wird zerstört), oder ob es sich um eine "Zielline" handelt -> Level fertig.

Augabenverteilung

Christian Blackert

- Gamestate (Aufrufe der einzelnen Spiel-Elemente und Menüs)
- Spielstart (Splash-Screen)
- Levelauswahl (Menü)
- Sound

Stefan Reidlinger

- Grafik (Design)
- Levelaufbau (Einlesen der Leveldefinition + Zeichnen von Sprites und Tiles)
- Bilden der Datenstruktur (Tiles + Vektoren) als Basis für Physik

Claus Liebenberger

- Spielablauf innerhalb des Levels
- Physik

Projektplan

15.05.2009: Fertigstellung Projektplan, Abschluß Aufgabenverteilung für Implementierung

bis 02.06.2009: Spielumgebung (Menüs, Gamestate, Sounds) + Levelaufbau abgeschlossen
Ziel: Ein Level soll statisch komplett darstellbar sein

02.06.2009: Zusammenführung der von CB und SR entwickelten Module

bis 10.06.2009: Implementierung der Spielephysik; Vervollständigung der Grafik und Menükomponenten; eventuell Level Editor (optional)

10.06.2009: Finalisierung der Applikation für Präsentation

bis 12.06.2009: Erstellung Postmortem

12.06.2009: Präsentation, Abgabe