# Team und Thema

Multimediasystemprojekt: Minigames

Felix Wild (k12216177):

Ein Bild, das Menschliches Gesicht, Person, Kleidung, Porträt enthält.

Automatisch generierte Beschreibung

Email: K12216177@students.jku.at

Alvin Schmid (k12103133)



Email: k12103133@students.jku.at

Cornelius Engl (k12216183)



Email: k12216183@students.jku.at

Felix Wöß (k12206357)



Email: K12206357@students.jku.at

Project Beschreibung:

Entwickelung

Mario World 1-1 (Nes): Das Spiel beginnt mit einer Hub-Welt (einer Nachbildung des ersten Levels im beliebten Nes-Spiel Mario Bros). Jedes Rohr in diesem Level kann verwendet werden, um auf ein anderes Spiel zuzugreifen.

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

Automatisch generierte Beschreibung

Welt laden:  
Die Welt wird als .tmx-Datei gespeichert. Die Karten basieren auf Kacheln. Jede Karte enthält Grafikebenen und Objektebenen. Im Programm wird die Datei ausgelesen. Für jedes Objekt in den Objektebenen fügt das Programm das Objekt als Körper in die Welt ein. Die Physik dieser Körper wird bei jedem Zyklus des Programms berechnet.

Logik:   
Durch einen Kontakt-Listener ist es möglich zu erkennen, wann zwei Körper miteinander kollidieren. Wenn die Körper kollidieren, kann festgelegt werden, was bei der Kollision geschehen soll. Zum Beispiel, wenn der Spieler mit der Unterseite eines Blocks kollidiert, wird ein Sound im Format .wav abgespielt und der Spieler kann nicht mehr mit dem Blockkörper kollidieren. Außerdem verschwindet die Grafik des Blocks in der Grafikebene. Das Gleiche passiert, wenn man einen Münzblock trifft, aber der Spieler kann weiterhin mit dem Block kollidieren und die Grafik ändert sich anstatt zu verschwinden.

Sprite:   
Der Sprite verwendet einen Körper und deren Grafik wird an der Position des Körpers gezeichnet. Die Animation wird in einer Schleife abgespielt, abhängig von der Geschwindigkeit des Sprite-Körpers. Wenn er sich in positiver x-Richtung bewegt, wird die Laufanimation abgespielt, wenn er sich in negativer x-Richtung bewegt, wird die Animation umgekehrt. Beim Springen wird die Sprunganimation abgespielt. Die Animationen bestehen aus einer Reihe von Bildern, die in einer bestimmten Abfolge abgespielt werden

Ein Bild, das Muster, Pixel, Farbigkeit, Screenshot enthält.

Automatisch generierte Beschreibung

pong

dinorunner

Susjump:

Der Spieler steuert den roten Imposter, der automatisch nach oben springt. Durch drücken der Pfeiltasten auf der Tastatur kann der Spieler die Bewegung des Imposters in horizontaler Richtung steuern. Der Imposter springt von Plattform zu Plattform, um Punkte zu sammeln.

Punktevergabe:

Normale Plattformen: Auf dem Bildschirm sind verschiedene Plattformen verteilt, die dem Imposter als Sprungbretter dienen. Jedes Mal, wenn der Imposter eine Plattform erfolgreich erreicht, erhält der Spieler 50 Punkte.

Brechende Plattformen: Es gibt auch spezielle Plattformen, die nach einem einzigen Sprung brechen. Diese Plattformen erfordern ein präzises Timing und Geschicklichkeit, um sie zu überwinden. Wenn der Imposter auf einer brechenden Plattform landet, erhält der Spieler 200 Punkte.

Aufgaben (Tasks): Zusätzlich zu den Plattformen gibt es im Spiel Collectibles, die der Imposter einsammeln kann. Diese Aufgaben werden in Form von Papierzetteln dargestellt. Jedes eingesammelte Aufgabenobjekt gibt dem Spieler 1000 Punkte.

Spielende:

Das Spiel endet, wenn der Imposter eine Wand oder den Boden berührt. Der Spieler sollte versuchen, den Imposter so hoch wie möglich springen zu lassen, um einen hohen Highscore zu erzielen. Der derzeitige Score wird im Spiel angezeigt und kann mit anderen Spielern verglichen werden.  
Da das Spiel endlos ist kann der Spieler kein Ziel erreichen, sondern einfach nur versuchen so lange wie möglich zu überleben.

Tileset: <https://aske4.itch.io/free-space-station>

(Das Tileset wurde um das doppelte vergrößert. Jeweilige Änderungen sind von Felix erstellt worden.)

JumpKing:

Der Spieler steuert eine Figur, welche sich mit den Pfeiltasteb steuern lässt. Diese Figur findet sich in einer Map wieder welche um die 240 Blöcke vertikal nach oben geht. Diese Map ist in drei Teile unterteil. Der Anfangsbereich ist ein Dungeon oder ein Keller, aus welchem der Spieler entkommen muss.

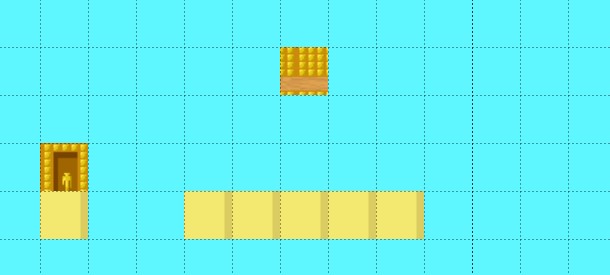


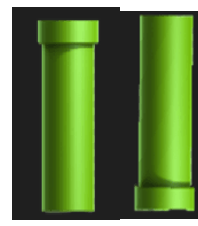
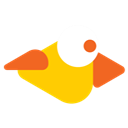
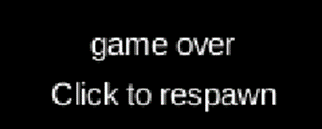
Es ist nicht möglich zu sterben, trotzdem sollte man nciht wieder herunterfallen, da es ebenso wenig Savepoints gibt.

Nachdem der Spieler dem Dungoen entkommen ist, kommt er in die Oberwelt. In dieser gibt es statt Ketten und Moos, Bäume und Sträucher.



Nachdem der Spieler die Oberwelt überstanden hat, kommt er in den dritten und letzten Abschnitt der Map. Hier steigt der Spieler in den Olymp auf. Am Ende dieses Abschnittes wartet der Ultimative Preis, das Ziel des ganzen Minigames. Ein Block aus reinem Gold.



**Mini-Game Flappy Bird**   
Alle Wege führen nach Rom. Ein guter Ansatz um ein Spiel wie Flappy Bird anzufangen, ist einen Vogel Charakter zu erstellen. Die erstellte Datei als .png abspeichern, um Transparenz zu gewährleisten. Nun wird die Größe des Vogels auf eine vernünftige Größe festgelegt, weder zu groß noch zu klein. Als Nächstes müssen wir eine y-Variable definieren, um die aktuelle vertikale Position des Vogels zu speichern. Eine x-Variable ist nicht erforderlich, da sich ja der Vogel nicht nach links oder rechts bewegen wird –> das übernehmen die grünen Pipe-Rohre. Wenn sich der Vogel nach oben und unten bewegen kann, können wir zum nächsten Schritt übergehen.   
Automatisierung der Vogelbewegung:   
Dazu mussten drei wichtige Elemente in unser Programm implementiert werden, und zwar die Schwerkraft, Benutzerinteraktion und Begrenzung.   
Um die Schwerkraft zu erzeugen, muss nur die zuvor definierte Y-Variable kontinuierlich um eine logische Zahl erhöhen, um den Vogel scheinbar zum Boden zu bringen. Die einzige Möglichkeit, dass Benutzer mit dem Vogel interagieren, besteht darin, eine bedingte Anweisung zu verwenden.   
Wenn Benutzer etwas tun, in diesem Fall ein Mausklick, bewegen wir den Vogel nach oben, indem wir die y-Variable um einen Wert verringern, der größer ist als die Schwerkraft. Darauf wurde für die Jump Animation dem Benutzer mit dem Drücken der linken Maustaste eine Interaktion gegeben und zusätzlich eine Sound Jump Effekt implementiert.   
Anschließend die Grenzen des Vogels setzen. Offensichtlich möchten wir nicht, dass der Vogel oder Teile des Vogels wegen zu hoher Flughöhe oder zu niedrigem Fall zu sehen ist. Um die Bewegungen des Vogels zu begrenzen, verwenden wir eine weitere bedingte Anweisung, um sicherzustellen, dass die y-Variable des Vogels größer als 0 und kleiner als die maximale Höhe des Bildschirms ist.   
Um es nicht langweilig zu machen, sollte unser Spiel die Konzentration und Fähigkeiten des Spielers herausfordern. Daher fügen wir unserem Programm Hindernisse hinzu wie diese grünen Rohre. Beim Spielen des Spiels fällt auf, dass der Vogelwahrscheinlich tausende von Rohren durchquert hat. Tatsächlich werden jedoch nur zwei Rohre benötigt. Beide Rohre werden in einem Zyklus platziert. Offensichtlich benötigen wir einige Variablen, um wichtige Werte wie die x-Position, die y-Position und die Höhe beider Rohre zu speichern. Anschließend animieren wir die Rohre und setzen sie in einen Zyklus. Hier kommt nun die Ergänzung. Wenn ein Rohr den oberen linken Teil des Bildschirms erreicht, anstatt die x-Variable direkt auf die Bildschirmbreite zu ändern, müssen wir die y-Position dieses Rohrs zufällig gestalten, die Höhe entsprechend anpassen und dann die x-Variable ändern, um das Rohr zu bewegen. Dadurch können die Spieler denken, dass jedes Rohr, das der Vogel passiert, einzigartig ist, was das allgemeine Spielerlebnis verbessert. Da es in einer Endlosschleife ist, ist leider der Tod des Vogels unvermeidlich. Zu jedem Anfang gehört immer ein Ende. Wir müssen unser Spiel irgendwie beenden, da niemand ein endloses Spiel spielen möchte. Definiert wird zunächst eine neue Variable, um zu überprüfen, ob das Spiel vorbei ist. Setzen wir den anfänglichen Wert dieser Variablen auf falsch. Als nächstes müssen wir erkennen, ob der Vogel auf eine grüne Pipe kollidiert. Es ist eine Funktion erforderlich, um das Spiel zu stoppen. Jede Kollision sollte diese Funktion auslösen. Wenn der Vogel ein Hindernis trifft, setzen wir den Wert der Variable "isGameOver" auf true. Beendet wird das Spiel mit einer Crash-Audiofile, wo sofort ein Popup erscheint und ebenso die Hintergrundmusik abgebrochen wird.  
  


Quellen:

https://www.youtube.com/watch?v=a8MPxzkwBwo&list=PLZm85UZQLd2SXQzsF-a0-pPF6IWDDdrXt&index=1

<https://github.com/crashinvaders/gdx-texture-packer-gui>

https://www.mapeditor.org

<https://www.spriters-resource.com/nes/supermariobros/sheet/52571>

https://o-gabrielvidal.itch.io/goldencity

https://pixakqrsho.itch.io/grass-and-dirt-2d-platformer-tileset

https://tilation.itch.io/multi-size-rocky-grass-tileset

https://ruskom.itch.io/moon-tileset

https://cardinalzebra.itch.io/dungeon-tiles-1

https://tilation.itch.io/multi-size-mythical-dungeon-tileset

https://oddpotatogift.itch.io/32x32-platformer

https://schwarnhild.itch.io/basic-tileset-and-asset-pack-32x32-pixels

https://gtajima.itch.io/2d-platformer-tiles-castle

https://marcon22.itch.io/platformer-tileset