**Android Studio Projekt**

Das Android Studio Projekt hat damit begonnen das wir die Aufgabenstellung von Ausbilder Fabian Joßberger erhalten haben, in zweier Gruppen uns eine Projektidee auszudenken und einen IHK Konformen Antrag an Ausbilder Fabian Joßberger zu stellen. Wir haben uns Gedanken zu verschiedenen Spielkonzepten gemacht und sind auf den Entschluss gekommen, als Projekt ein Spiel zu programmieren in dem es möglich ist auf einer Insel mit einem Player und Joystick sich zu bewegen und in vordefinierten Feldern Samen zu pflanzen, die mit der Zeit wachsen und welche am Ende geerntet werden, um verschiedenstes Gemüse zu erhalten. Dieses Gemüse sowieso weitere Items wie die Samen werden in einem Inventar gelagert und Samen können daraus per Drag and Drop auf die vordefinierten Felder gezogen werden, um diese anzupflanzen. Weitere Funktionalitäten wären die Uhr sowie als auch der Tageszähler, das Level System mit dem weitere Felder freigeschaltet werden können, ein Händler bei dem Samen gekauft sowie Erträge verkauft werden können, ein Währungssystem mit dem beim Händler gekauft und verkauft werden kann, Borders, welche es ermöglichen, dass der Spieler hinter Objekte wie Bäume oder Häuser laufen kann und verhindern, dass der Spieler aus der Welt laufen kann und eine Database, die den Spielstand speichert und beim Starten der wieder Aufruft.

Für die Entwicklung unserer Android-Spiel-App haben wir uns bewusst für das Framework LibGDX entschieden. LibGDX eignet sich hervorragend für mobile Spiele, da es speziell auf Performance und Effizienz ausgelegt ist. Besonders im Hinblick auf ressourcenschonende Darstellung und flüssige Animationen auf mobilen Geräten überzeugte uns das Framework. Ein großer Vorteil von LibGDX ist die einfache Projektstruktur, Zugriff auf Android-Funktionen und einer gut dokumentierten API für Eingaben, Audio und Grafik. Dadurch konnten wir Eingabeinteraktionen über den Touchscreen problemlos umsetzen und gleichzeitig ein flüssiges Spielerlebnis gewährleisten. Auch der Einsatz externer Tools wie Tiled zum Erstellen von Karten und Leveln ließ sich problemlos mit LibGDX kombinieren, wodurch wir effizient Inhalte erstellen und ins Spiel integrieren konnten. Insgesamt war LibGDX die ideale Grundlage für unser Android-Spiel, da es eine ausgewogene Mischung aus technischer Kontrolle, Flexibilität und praktischer Umsetzbarkeit für mobile Plattformen bietet.

**Main**

Die Main Klasse ist wie der Startknopf vom Spiel. Sobald man das Spiel öffnet und die Android App gestartet wurde, springt das Programm direkt in die Main Klasse rein. Von hier aus wird alles aufgebaut. Als erstes wird in der create Methode geschaut, ob ein Datenbank-Service vorhanden ist, den bekommt die Klasse über den Konstruktor übergeben. Je nachdem ob man am PC ist oder auf Android wird dann ein anderer Service genommen. Danach wird sofort gesagt dass der StartScreen aufgerufen werden soll, das ist der allererste Bildschirm den der Spieler sieht.

In der Main Klasse wird aber nicht nur einfach gestartet sondern hier wird auch festgelegt, wie andere Screens wie GameScreen oder MainMenuScreen geladen werden. Das heißt man kann von überall her sagen main.setScreen(...) und dann zeigt das Spiel genau den Bildschirm. Außerdem gibt es Methoden um den aktiven Screen zu verändern, also zum Beispiel wenn man vom Menü ins Spiel oder vom Spiel ins Inventar möchte.

Ganz wichtig ist hier auch das dispose, das sorgt am Ende dafür, dass alles was nicht mehr gebraucht wird gelöscht wird aus dem Speicher, was bei mobilen Geräten wie Handys wichtig ist damit die App nicht langsam wird oder abstürzt. Was man in dieser Klasse auch sieht ist, dass sie die Verbindung zu den Datenbank Klassen hält, damit auch andere Klassen später auf Spielstände oder das Inventar zugreifen können. Also im Grunde: alles geht von hier los und alles endet auch wieder hier, wenn man so will ist die Main Klasse das Zentrum vom ganzen Spiel.

**StartScreen**

Die StartScreen Klasse ist die erste Ansicht die man zu sehen bekommt, wenn man das Spiel startet. Bevor überhaupt ein Menü kommt oder man sich bewegen kann oder irgendwas klicken kann, wird dieser Screen aufgerufen. In der Klasse werden schon einige Dinge vorbereitet, zum Beispiel wird ein Text angezeigt der "Farm Life" steht oder wie das Spiel heißt und im Hintergrund wird Musik gestartet, die dann durchläuft bis man ins Menü kommt. Die Schrift wird mit einer BitmapFont gemacht, die extra noch skaliert wird damit sie größer ist und besser lesbar auf dem Handy. Der Hintergrund ist ganz einfach ein schwarzes Rechteck, das über den gesamten Bildschirm gemalt wird.

Was hier im Code auch passiert ist, dass ein Timer mitläuft. Der zählt die Sekunden hoch und sobald zum Beispiel 5 Sekunden vergangen sind oder der Nutzer irgendwo tippt (also der Touchscreen benutzt wird), dann wird automatisch in den MainMenuScreen gewechselt. Dieser Wechsel passiert über die setScreen Methode, die Teil von LibGDX ist. Außerdem wird schon in der render Methode alles vorbereitet und immer wieder neu gemalt, also quasi das Herz vom StartScreen. Die Methode dispose sorgt noch dafür dass der Speicher wieder freigegeben wird, wenn man diesen Screen verlässt.

Also insgesamt passiert in dieser Klasse schon einiges im Hintergrund, was der Spieler gar nicht direkt merkt, aber was wichtig ist damit das Spiel flüssig startet und ein guter erster Eindruck entsteht. Und durch die Timer und den Screen Wechsel fühlt sich das Ganze wie ein kleines Intro an, was halt wichtig ist damit sich das Spiel auch wie ein echtes Spiel anfühlt.

**MainMenuScreen**

Der MainMenuScreen ist der Bildschirm den man nach dem StartScreen zu sehen bekommt, also quasi das Hauptmenü vom Spiel. Hier entscheidet der Spieler was er machen möchte, ob er ein neues Spiel anfangen will, ob er vielleicht ein gespeichertes Spiel laden will oder irgendwann vielleicht sogar Optionen ändern möchte. In dieser Klasse wird eine Stage benutzt, das ist eine Art Container in LibGDX für alles was auf dem Bildschirm ist, und dann werden da Buttons drauf gesetzt.

Jeder Button hat eine bestimmte Funktion. Zum Beispiel gibt es einen Start-Button der in den GameScreen führt, also dahin wo das eigentliche Spiel losgeht. Wenn man diesen Button klickt, wird mit setScreen der Bildschirm gewechselt. Hier arbeiten wir mit Skins, die festlegen wie die Buttons aussehen sollen, also Schrift, Farbe, Größe usw. Die Stage wird in der render Methode immer wieder neu gezeichnet und verarbeitet auch Eingaben, das heißt das Spiel weiß genau, wann der Nutzer wo drauf tippt.

Insgesamt ist der MainMenuScreen nicht so lang von der Logik her, aber ziemlich wichtig, weil er das ist was der Spieler nach dem Start sieht und was entscheidet wie schnell man ins Spiel findet. Auch hier ist wieder dispose dabei damit alles ordentlich aufgeräumt wird wenn man das Menü verlässt. Es ist also ein kleiner aber wichtiger Teil der den Übergang vom Intro ins eigentliche Spiel darstellt.

**Map:**

Die Map besteht aus einer TiledMap die in der App Tiled aus Assetpacks, also aus dem Internet heruntergeladenen, vorgefertigten Bilder mit verschiedensten Zuständen, Blöcken und Charakteren , erstellt wurde und verschiedenste Ebenen für das Design enthält. Die Ebenen teilen sich auf in Kachelebenen und Objektebenen. Kachelebenen beinhalten das Design und Objektebenen beinhalten Formen mit denen Funktionalitäten geschrieben werden können. In der Klasse Map laden wir die Map mithilfe der TiledMap Klasse von LibGDX die aus dem assets Ordner des Projekts die Map raus zieht und mit den Funktionalitäten der TiledMap Klasse in unseren Code lädt. Daraufhin wird die Map in 3 Ebenen unterteilt, die Grundebenen die geladen werden, dann wird der spätere Spieler geladen, daraufhin die Layer die über dem Spieler liegen sollen und zum Schluss noch die Objektebenen die für die Borders zuständig sind. In Map klasse befindet sich auch eine Funktion für die Borders die durch die Objektebenen geht, die Formen raus zieht und prüft ob die Form, die zukünftige Position vom Spieler beinhaltet. Je nachdem was die Form für eine Funktion hat wird die Bewegung des Spielers gestattet oder nicht. Zum Beispiel soll die Map Border Form die Position des Spielers beinhalten, aber See Border Form darf die Position des Spielers nicht beinhalten. Die Map wird zu beginn reingezoomt, damit sie ein benutzerfreundliches und gut ausschauendes Erlebnis liefert und die Map wurde deshalb auf eine Welt Kamera gesetzt, die es ermöglicht die Map, wie mit einer Drohne in Top View zu überblicken. Die Welt Kamera bewegt sich mit jedem Schritt des Players mit.

**Joystick:**

Der Joystick besteht aus 2 Kreisen, einem kleinen der für die Bewegung des Spielers, sowie der dazugehörigen Weltkamera zuständig ist und einen großen, der die grenze für den kleinen Kreis setzt. Der Joystick wird aber nicht auf der Weltkamera abgebildet, sondern besitzt eine eigene Kamera die einen Sichtbereich des Handys hat. Vorteil einer solchen HUD-Kamera ist, dass die Elemente dieser Kamera dauerhaft auf derselben Position angezeigt werden. Der Joystick beinhaltet, dann noch Logik, die erkennt, wenn der Bildschirm berührt wird und wo der Bildschirm berührt wird, weil die Position des Joysticks soll auf die Position der Berührung gesetzt werden, solange die Berührung in der linken Hälfte des Bildschirms passiert. Während dem Berühren und Verwenden des Joysticks ist die Position des Joysticks vorerst fest und die Position des kleinen Kreises wird auf die Position der weiteren Berührung gesetzt, solange diese im großen Kreis ist. Wenn die Berührung den großen Kreis verlässt, wird der Winkel zum Mittelpunkt des großen Kreises berechnet und auf den Radius des großen Kreises zurückgesetzt. Durch die Distanz vom kleinen Kreis zum großen Kreis wird die neue Position der Welt-Kamera und des Spielers berechnet und an die Map Klasse zum Testen wegen der Borders übergeben. Die letzte Funktionalität der Joystick Klasse ist es mithilfe eines ShapeRenders, welcher es einfach ermöglicht auf einer Kamera frei Formen zu zeichnen, den Joystick zu malen. Die Funktionalität des Joysticks übergibt auch noch die Richtung in die sich bewegt wird oder in die sie zuletzt bewegt wurde. Diese beiden Funktionalitäten werden dann in der GameScreen klasse verwendet.

**Player:**

In dieser Klasse werden aus einem SpriteSheet, ein Koordinatensystem gemacht

**Clock:**

**Felder:**

**Item:**

**Pflanzen:**

**Hitboxes:**

**Gamescreen:**

**Inventory:**