Crack Tomb
Technical Design Document

Teammitglieder:
Gabriel Moczalla (Teamleiter)
Jannick Knechtel
Anne Döbler

Kontaktinformationen:
Gabriel Moczalla – gabrielmoczalla@st.ovgu.de
Jannick Knechtel –
Anne Döbler – annedoebler@st.ovgu.de

Inhaltsverzeichnis:

1.Technische Mindestanforderungen:

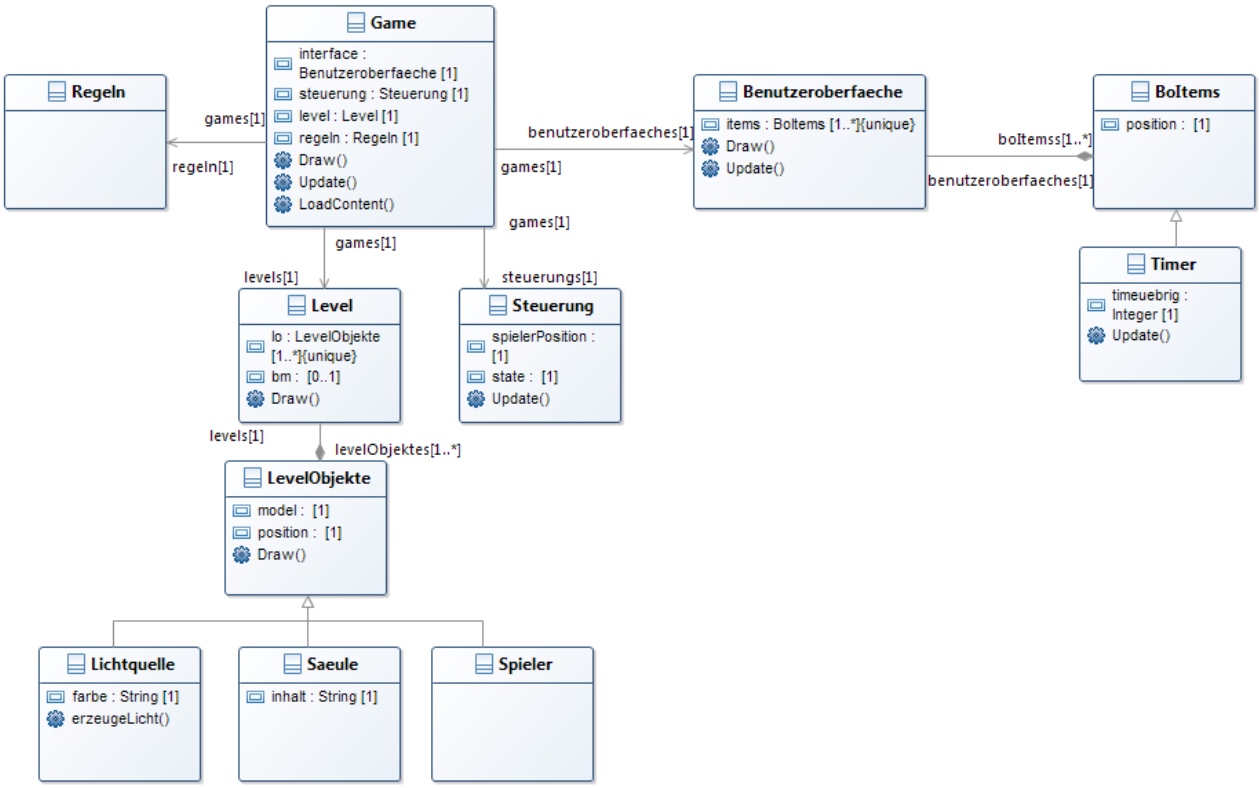
Zur Zeit der Entwicklung werden Anforderungen an die Systeme der Entwickler, also uns gestellt. Benötigt werden auf den Systemen eines der Betriebssysteme von Windows. Hierunter zählen Windows XP, Windows Vista, Windows 7 oder Windows 8. Des Weiteren benötigen die Systeme das Framework XNA 4.0. Dieses fordert wiederum eine Entwicklungsumgebung. Von XNA 4.0 unterstütze Entwicklungsumgebungen sind Microsoft Visual Studio C# Expression, Microsoft Visual Studio 2010 Standard oder Microsoft Visual Studio 2010 Professional. XNA 4.0 fordert nicht nur sondern installiert automatisch das Fraumework Microsoft .NET 4.0 und die DirectX-Laufzeit. Dies sind alle Voraussetzungen um mit XNA 4.0 arbeiten zu können. Jedoch werden noch weitere Tools benötigt, um ein tiefes Arbeiten zu ermöglichen. Diese werden im Punkt „Zusätzliche Tools“ erläutert.
Um das fertige Spiel spielen und auch testen zu können, werden Anforderungen an die Spiel- bzw. Testsysteme gesetzt. Diese wären wieder ein Betriebssystem von Windows, wie oben genannt. Die Grafikkarte des Systems sollte außerdem mindestens das Shader Model 1.1 beherrschen und auch DirectX 9.0c unterstützen.

2.Softwarearchitektur:

Zum Thema Softwarearchitektur sieht man, im untenstehenden, Modell 1, das Klassendiagramm, und Modell 2, das Use-Case-Diagramm. Im Use-Case-Diagramm sieht man nahezu alles was im Menü und im Spiel passieren kann. Im Klassendiagramm sieht man eine ungefähre Programmierstruktur. Im folgenden wird die Programmierstruktur weiter und genauer erklärt. Wie in „Struktur des Spiels“ (GDD) zu erkennen ist gibt es einen Haupt-Menü-Bildschirm mit dem das Spiel starten wird. Auf diesem sind mehrere Buttons, die einmal zu den Credits, und einmal zur Levelauswahl führen. Auf dieser ist wieder ein Button der zur lokalen Rangliste führt, die auf dem Medium auf dem gespielt wird, gespeichert ist. Von der Levelauswahl kann man logischerweise die Level auswählen. Anfangs ist nur das erste Level auswählbar, die anderen sind gesperrt, und das nächste Level wird erst mit erfolgreichen Abschluss des vorhergehenden Levels freigeschalten. Technisch stellt sowohl das Hauptmenü, als auch Credits, Levelauswahl und Rangliste eine eigene Klasse im Code dar, mit ihren eigenen Objekten.
Auch jedes Level ist eine eigene Klasse. Für jedes Level werden alle in dem Level vorkommenden Level-Objekte erstellt, sowie dem alle Objekte die der Spieler zum lösen des Levels benötigt, in sein Inventar eingefügt. Der Spieler spawnt an einer festgelegten Stelle. Mit dem Spawn beginnt auch die Uhr zu laufen.
Alle Level-Objekt sind jeweils wieder eine Klasse. Jedes Objekt hat mehrere Werte die seine Eigenschaften eindeutig bestimmten, und jedes Level-Objekt hat eine Textur oder ein Modell, durch welches es grafisch repräsentiert wird. Jedes Level-Objekt hat außerdem eine ID, die für die spätere Kollision benötigt wird. Das Level-Objekt „Wand“ besteht aus einem variablen X-Wert und einem Y-Wert für die Größe, und einem festen Z-Wert. Für die Position gibt es wieder variablen einen X-Wert, einen Y-Wert und einen Z-Wert die die Position einer bestimmten Ecke der Wand markieren. Das Level-Objekt „Wand mit Loch“, hat genau die selben Werte, nur eine andere ID, und eine andere Textur. Das Level-Objekt

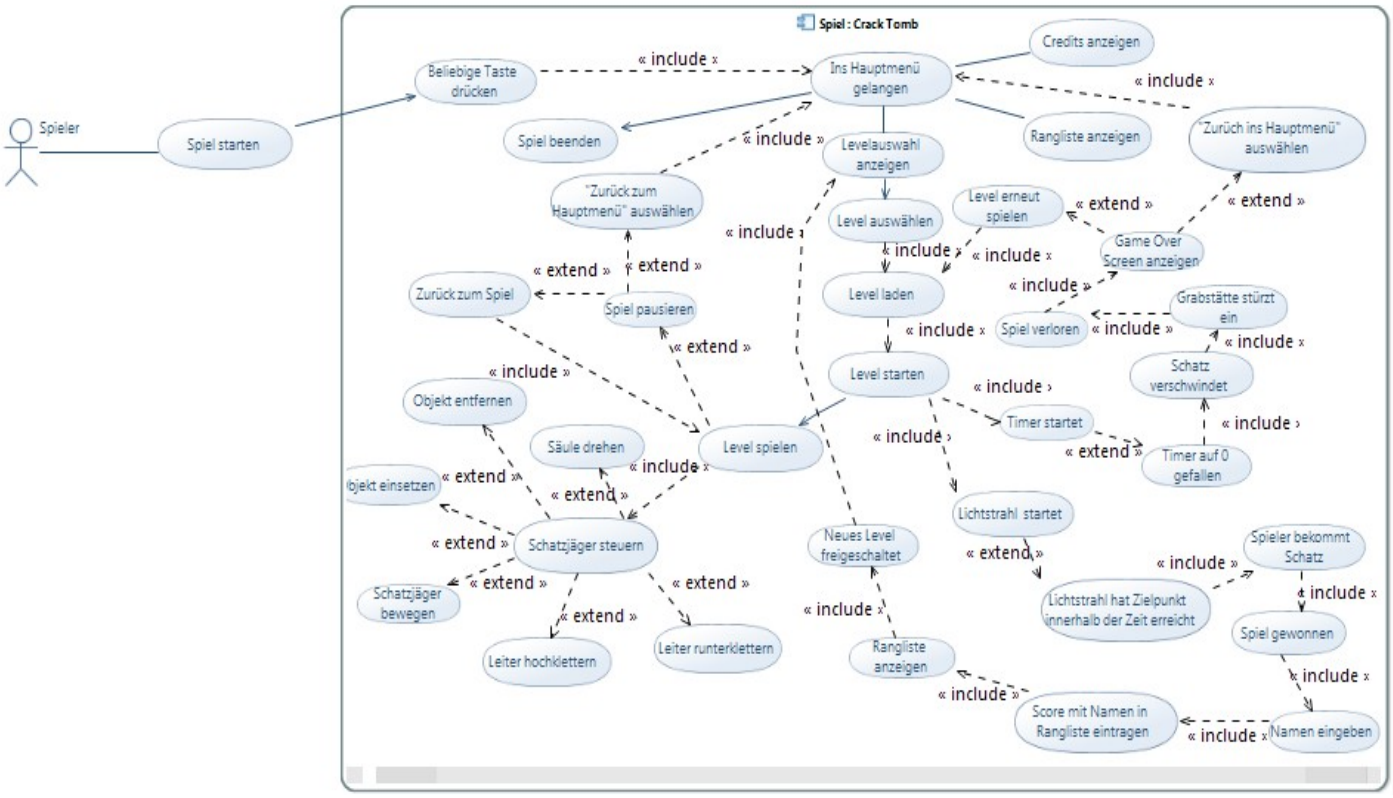
„Säule“ ist immer gleich groß, daher benötigt es nur einen X-Wert, einen Y-Wert und einen Z-Wert für die Position. Zusätzlich hat die Säule eine Funktion mit der der Spieler ein Objekt in einer frei wählbaren Ausrichtung in die Säule einsetzten kann, oder es wieder entfernen kann. Säulen haben außerdem einen Bool-Wert fest/ nicht fest. Das Objekt „Leiter“ hat auch immer die selbe Größe und benötigt nur 3 Werte für seine Position. Die Leiter wird immer an einer Wand platziert. Das Level-Objekt „Tür“ hat ebenfalls immer die gleiche Größe, daher auch 3 Werte für die Position. Die Tür wird immer zwischen 2 Wand-Objekten platziert. Die Tür hat zusätzlich eine „Tür-NR“ die für die Schalter wichtig ist, und einen Bool-Wert für offen/geschlossen. Die „Lichtbarriere“ hat auch 3 Positions-Werte und keine variable Größe. Dazu hat sie einen „Farbwert“ der als 6-stelliger Binärcode gespeichert wird (z.B. 001010). Das das Spiel mit 6 verschieden Farbkristallen arbeitet wird der Anteils jeder einzelnen Farbe durch ein Bit gespeichert. Die selbe Technik verwendet auch der Lichtstrahl später. Der „Schalter“ hat auch immer die selbe Größe, daher nur 3 Positions-Werte. Jeder Schalter hat einen Bool-Wert aktiviert/deaktiviert. Optional kann ein Schalten einen „Farbwert“ wie die Lichtbarriere haben. Jeder Schalter zeigt auf eine oder mehrere Türen, die er öffnet oder schließt. Die „Truhe“ hat auch wieder nur 3 Positions-Werte und ein wie der Spieler ein Inventar mit ein oder mehreren Items. Die Sonderobjekte Start-Punkt und End-Punkt haben auch nur 3-Positionswerte.
Alle Objekte können durch einen einfachen Konstruktor, der nur mit den benötigten Werten befüllt werden muss, erstellt werden. Die jeweiligen Level können dann „gehardcoded“ werden, nachdem die vorher irgendwo konstruiert worden.
Die Spieler-Items sind ähnlich aufgebaut. Alle haben eine eigene Textur und ID. „Spiegel“ und „Doppelseitiger Spiegel“ haben keinen weiteren Wert. Der Farbkristall hat einen Farbwert der binär kodiert wird wie bei der Farbbarriere. Die „Splitkristalle“ werden mit einem 5-stelligen Binärwert codiert. Jedes Bit steht für eine der möglichen Splitrichtungen Links, Rechts, Geradeaus, Oben, Unten.
Diese werden auch in jedem Level in das Inventar des Spielers, oder die Truhen, hardgecoded.
Der Spieler kann sich mit WASD bewegen. Der Spieler kann nicht durch geschlossene Türen, Wände oder Säulen laufen. Kollidiert er mit diesen bleibt er stehen. Läuft der Spieler gegen eine Leiter mit „W“ klettert er an ihr nach oben. Wenn er das nicht tut und keinen Boden unter den Füßen hat sinkt er nach unten. Das Inventar das Spieler wird ständig rechts angezeigt. Mit linken Mausklick kann der Spieler auf Truhen und Säulen zugreifen. Dort kann er per Drag-and-Drop die Items in oder aus den Säulen nehmen. Wird in eine Säule ein Spiegel oder ein Split eingeführt hat der Spieler zusätzlich die Option diesen per Mausklick auf Pfeile zu drehen um seine Ausrichtung zu verändern.
Der „Lichtstrahl“ spawnt in seinem Startpunkt und breitet sich linear in die vorgegebene Richtung aus. Anfangs hat er die Farbe Weiß (000000). Kollidiert der Strahl mit irgendetwas wird die ID des Objekt mit dem er kollidiert überprüft und je nachdem auf was der Strahl getroffen ist entweder ein/mehrere neue Strahle erzeugt, oder der Strahl gestoppt. Trifft der Strahl auf den Endpunkt kommt der Level-geschafft-Bildschirm. Zur grafischen Darstellung des Lichtstrahl wird ein Partikelsystem oder ein sich ausdehnender Zylinder verwendet.
Bei der Kamera gibt es die Option der First-Person Kamera oder einer festen Kamera die von oben auf das Level guckt und die Ebene sieht in der sich der Spieler aktuell befindet.

Klassendiagramm:



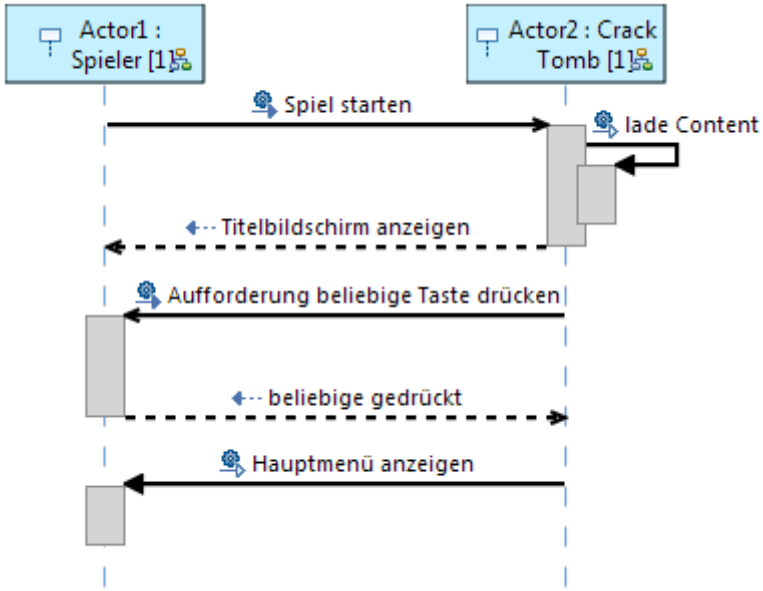
Modell 1

Use-Case-Diagramm:

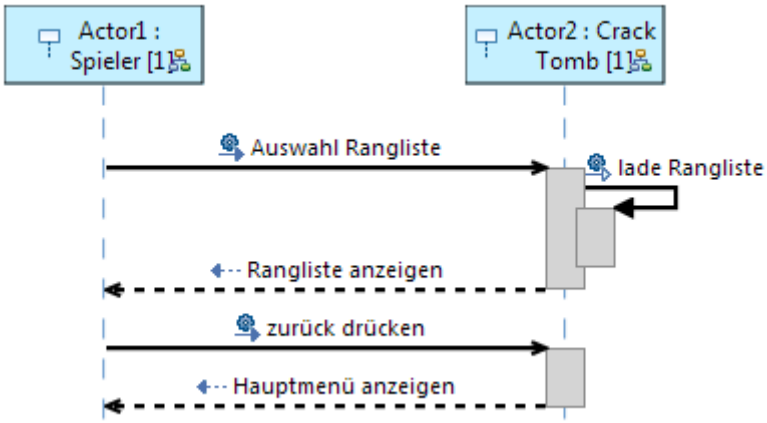


Modell 2

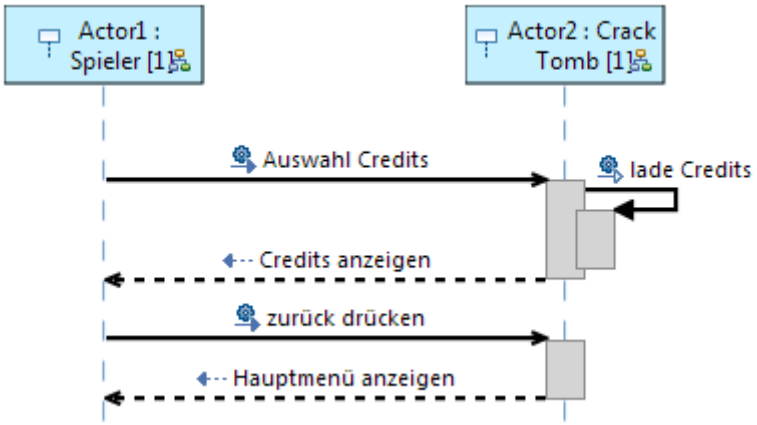
Sequenzdiagramme:



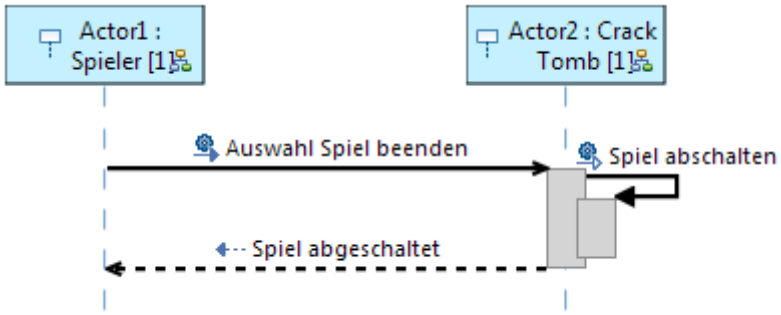
Titelbildschirm-zu-Hauptmenü



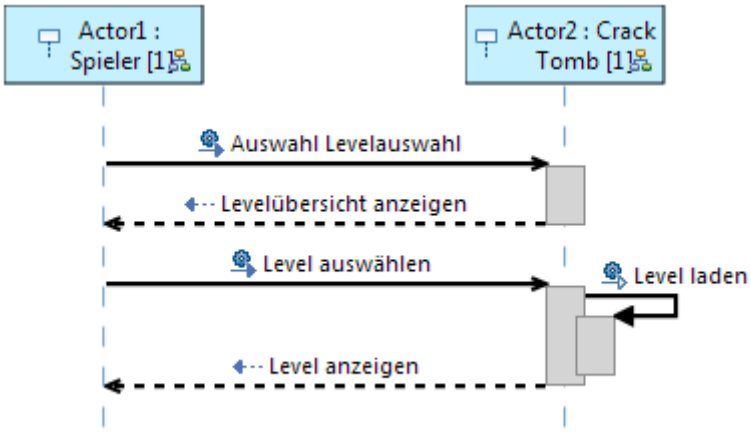
Hauptmenü-zu-Rangliste



Hauptmenü-zu-Credits



Spiel-beenden



Hauptmenü-zu-Level

3.Bibliotheken:

Bibliotheken die wir nutzen werden, beschränken sich für den Anfang auf die Standartbibliotheken des XNA 4.0 Frameworks. Ansonsten werden größtenteils eigene Bibliotheken implementiert. Beispielsweise wird die Implementierung des Lichtstrahls höchstwahrscheinlich selbst geregelt.

4.Zusätzliche Tools:

Zusätzliche Programme, die wir verwenden werden,
sind die kostenlose Programme Blender, Gimp und Audacity.
Blender ist eine 3D-Software, mit der man 3D-Objekte modellieren und rendern kann. Außerdem können Animationen erstellt werden. Hierbei ist zu beachten, dass man das Format .fbx einhält.
Gimp ist ein pixelbasiertes Bildbearbeitungsprogramm. Dieses wird in diesem Projekt ausschließlich für die Erstellung der Texturen für die Objekte und Menüs dienen. Die Texturen werden das Format .png besitzen.
Audacity ist ein Audio-Editor zum Aufnehmen, Bearbeiten und Abspielen von Audio-Dateien. Eines der Formate .wma, .mp3 oder .wav soll eingehalten werden.

5.Projektmanagement:

Um zu lange Anfahrtszeiten zu vermeiden erfolgt die Kommunikation über den Anbieter „Skype“. Dort wird jeden Montag um 17:00 ein Meeting abgehalten, in dem der wöchentliche Fortschritt besprochen wird und die neuen Aufgaben verteilt werden. Für das Projekt wird ein öffentliches GitHub verwendet damit alle gleichzeitig am Projekt arbeiten können. Wir verwenden kein komplexes IT-Projektmanagement-System wie „Scrum“ oder etwas ähnliches, da dies für ein 3-Mann-Projekt den Aufwand nur erhöhen würde. Wir verwenden auch kein einfaches Modell wie das „Wasserfall“-Modell im dem Sinne, da wir das hier nicht für sinnvoll erachten. Statt dessen verwenden wir ein „freies Projektmanagement“. Aufgaben werden je nachdem was jeder am besten kann aufs Team verteilt.

6.Arbeitspakete:

1. *Wie wird die Lichtquelle funktionieren?*
   Hier soll eine Recherche stattfinden, wie die Lichtquelle im Spiel umgesetzt wird. Insbesondere soll darauf eingegangen werden, wie man den Lichtstrahl umlenken und stoppen kann.
2. *Wie wird die Struktur der Level sein (Funktionsweise u. Technische Umsetzung)?*
   Hier soll recherchiert werden, wie die Level umgesetzt werden. Wichtig hier ist, dass man die Level einfach erstellen und auch laden kann.
3. *Wie funktioniert das .fbx Format?*
   Eine Recherche über dieses Format soll klären, wie man mit ihm arbeitet und wie man diesen erfolgreich und effizient in XNA nutzen kann.
4. *Klassendiagramm (Struktur) umsetzen.*
   Eine Umsetzung der reinen Struktur soll vollzogen werden. Dies bedeutet, dass Klassen deklariert werden sollen, also prinzipiell wird das Klassendiagramm in XNA übertragen, sodass eine Arbeit der anderen Gruppenmitglieder ermöglicht wird.
5. *Menüs designen.*
   Es soll das Design der Menüs erstellt werden. Eine Umsetzung jedoch nicht.
6. *Menüs umsetzen.*
   Die vorher designeten Menüs sollen nun in der Struktur eingebettet werden. Hierbei muss das Design beachtet werden.
7. *Lichtstrahl umsetzen.*
   Der recherchierte Lichtstrahl soll nun in die Struktur eingebettet werden.
8. *Levels umsetzen.*
   Die Struktur für die Level soll umgesetzt werden.
9. *Level erstellen.*
   Level sollen so erstellt werden, dass diese ins Spiel geladen werden können und auch gameplaytechnisch sinnvoll sind. Zur Folge hat dieses auch Tests der Level.
10. *Modelle erstellen.*
    Modelle sollen erstellt werden zu den geforderten Konditionen.
11. *Animationen erstellen.*
    Animationen für die Level und den Charakter sollen erstellt werden.

12. *Titelbild gestalten.*
    Das Titelbild soll erstellt werden.
13. *Sounds recherchieren/auswählen.*
    Sound sollen erstellt und ausgewählt werden.