# V4T PROJEKT

# E-LEARNING KURS

# MODUL 2

## Kapitel 1 - Einführung in die Programmierung von Videospielen

### Verfügbare Technologien- Teil 1

Die Spieleentwicklung umfasst mehrere verschiedene Arbeits- und Wissensbereiche, von der künstlerischen Gestaltung bis hin zur Low-Level-Programmierung. Für jeden Bereich gibt es Hilfs- oder Entwicklungswerkzeuge und spezielle Technologien. Die meisten dieser Tools können unabhängig voneinander verwendet werden, was den Spieleentwicklerteams die Möglichkeit gibt, für einen großen Teil des Projekts getrennt zu arbeiten. In der Endphase der Entwicklung wird die gesamte Arbeit verschiedener Teams und Kreativer in das Endprodukt integriert. Um dies zu erreichen, verwendet das Spieleentwicklungsteam Software, die alle Elemente des Spiels übernimmt und das Endprodukt generiert. Diese Software ist in der Lage, Bilder, Sounds, Videos, zusammen mit einem Regelwerk oder einem Programm zu verwalten und das eigentliche Spiel zu generieren.

Im Wesentlichen ist die Erstellung eines Spiels Programmierung, wie bei der Erstellung jeder anderen Software auch. Mit der Weiterentwicklung der Software und der zunehmenden Spezialisierung entstehen jedoch Unterschiede zwischen Spielen und der allgemeinen Softwareentwicklung. Und die Werkzeuge, die zur Unterstützung von Softwareentwicklern eingesetzt werden, entwickeln sich weiter und werden dadurch spezialisiert. Häufig erstellen Softwarefirmen neue Werkzeuge, um bestehende Programm, Webentwicklungen, eingebettete Software usw. weiterzuentwickeln, die einen Sprung in Sachen Vereinfachung und Effizienz in diesem Bereich versprechen. Dies gilt auch für Spieleentwicklungswerkzeuge.

Im Gegensatz zu anderen Softwarebranchen reichen die Technologien zur Spieleentwicklung jedoch von sehr maßgeschneiderten bis hin zu sehr offenen Tools. Es gibt einige Tools, die verwendet werden können, um komplexe Spiele eines beliebigen Genres zu erstellen, während andere Tools das Ziel haben, eine ganz bestimmte Art von Spiel zu erstellen. Es gibt sogar einige Spiele, die mit den notwendigen Tools ausgestattet sind, um dieses Spiel um neue Charaktere oder Levels zu erweitern und so ein gemischtes Tool zur Spieleentwicklung zu schaffen.

Co-funded by the Erasmus+ Programme of the European Union
The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Darüber hinaus teilt sich die Spieleentwicklung einige Schlüsseltechnologien mit anderen Bereichen der Wissenschaft und der Informationstechnologien. Zum Beispiel ist die 3D-Bildgebung ein großer Teil der Spielrevolution, und die zugrunde liegenden Grundlagen sind auch die Grundlage für Computer unterstütztes Design und Computer unterstützte Produktion (CAD/CAM), Computergestützte Chirurgie, globale Kartenbetrachter (Google Earth), und so weiter.

### 1.2 – Verfügbare Technologien - Teil 2

Es gibt eine Liste von Informatikbereichen, die in direktem Zusammenhang mit der Spieleentwicklung stehen:

- Softwareentwicklung. Wie jede Computersoftware werden Spiele von EntwicklerInnen entwickelt, die qualifiziert und geschult sein müssen, um die neuesten Techniken der Softwareentwicklung anzuwenden zu können.
- Programmiersprachen. sind unerlässlich, um eine komplexe Software zu entwickeln.
- 3D- und 2D-Bildgebung. Spiele haben sehr hohe Ansprüche an die Software, wenn es darum geht, Bilder, Animationen, Bewegungen usw. anzuzeigen. Dies hat die Hardwarehersteller dazu veranlasst, leistungsstarke Grafikkarten zu entwickeln und zu bauen, die komplexe Szenen in einer kontinuierlichen und flüssigen Animation darstellen können. Infolgedessen sind einige Standards entstanden, die es Softwareherstellern ermöglichen, diese leistungsstarke Hardware zu nutzen, wobei OpenGL das universellste ist. Darüber hinaus wurde die Modellierungssoftware, die Menschen hilft, Objekte in einer dreidimensionalen Welt zu entwerfen, ausgereifter und einfacher zu bedienen.
- Maschinelles Lernen und künstliche Intelligenz. Das Spielen gegen den Computer erfordert, dass der/die EntwicklerIn das Spiel mit einer gewissen Logik programmiert, wie man den/die SpielerIn herausfordert. Normalerweise hat das Spiel "Schwierigkeitsgrade", um den Computer mit den Fähigkeiten des/ SpielerIn abzugleichen. Es gibt jedoch ständige Bemühungen, Spiele zu entwickeln, die sich automatisch an die SpielerInnen anpassen und lernen, wie man sie im Laufe der Zeit besiegt.

Darüber hinaus dürfen wir nicht jene Technologien, Wissen und Entwicklungen vergessen, jenseits der Programmierarbeit, die die Spieleentwicklung unterstützen.

- Kino, Comics und Grafikdesign. Spiele neigen dazu, die Charaktere, Landschaften und Umgebungen auf die atemberaubendste Weise darzustellen und versuchen, den/die SpielerIn durch ein kraftvolles Erlebnis zu bringen. Es gibt bekannte und aufkommende Techniken in der Filmproduktion, die für die Herstellung von Spielen wiederverwendet werden. Diese Techniken sind selbst eine Art Technologie, die SpieleentwicklerInnen erwerben müssen. Spiele spiegeln die Szenenaufnahme von Filmen wider.



Co-funded by the Erasmus+ Programme of the European Union
The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

- Psychologie. Aus einer bestimmten Perspektive ist das Spielen von Spielen wie das Ausbrechen aus der Realität. SpieleentwicklerInnen sind erfolgreich, wenn ihr Spiel fesselnd ist und der/die SpielerIn für kurze Zeit aus der Welt, in der er/sie tatsächlich lebt, ausbricht. Einige Spiele gehen weiter und zielen auf die Gefühle der SpielerInnen wie Trauer, Wut, Angst usw. ab. Spiele orientieren sich an der Psychologie, um diese Ziele besser zu erreichen.
- Sozialwissenschaften. Multiplayer-Spiele, insbesondere massive Multiplayer-Spiele, nutzen die Sozialisierung der SpielerInnen und nutzen die zusätzliche Motivation, die ein Spieler spürt, wenn er/sie weiß, dass er/sie gegen eine andere Person spielt. Das Spiel wird zu einem Instrument, um eine Person dazu zu bringen, (während des Spiels) andere zu treffen. Die Art und Weise, wie SpielerInnen interagieren, ist Thema vieler Studien. Menschen neigen dazu, Freunde zu finden und auch Feinde, Partien zu planen, etc. Es gibt einen Bereich in den Sozialwissenschaften, der mit Online-Massiv-Multiplayer-Spielen verbunden ist. Das Ergebnis ist wertvolles Wissen, das für neue Multiplayer-Spiele anwendbar ist.

## 1.2 Entwicklungsumgebung und Programmiersprachen – Teil 1

Entwicklungsumgebung oder "Integrierte Entwicklungsumgebung" (IDE) ist eine spezielle Software, die Entwicklern hilft, Software zu entwickeln. Die Basis-IDE beinhaltet einen Editor, einen Compiler und einen Debugger, und heutzutage auch mehrere andere nützliche Tools. Da Softwaretypen immer unterschiedlicher werden, werden auch IDEs immer unterschiedlicher. Daher gibt es eine breite Palette an IDEs, die auf die Softwareentwicklung in verschiedenen Bereichen ausgerichtet sind. IDEs zur Entwicklung von Spielen sind eine der spezialisiertesten, mit vielen Besonderheiten. Ein zentrales Element jeder Softwareentwicklung ist die verwendete Sprache. Einige IDEs sind für eine bestimmte Sprache ausgelegt, während andere für die Verwendung mehrerer Compiler oder Sprachen konfiguriert werden können.

IDEs für die Spieleentwicklung haben mehrere spezielle Funktionen, aber zwei sind von großer Bedeutung. Das erste ist die Integration mehrerer Tools, um verschiedene Teile eines Spiels zu produzieren (Grafik, Programmierung, Sounds, Networking, etc.), und das zweite große Feature ist die Spezialisierung auf die Spieleentwicklung, bei der die meisten Elemente eines Spiels vorkonfiguriert sind (Level, Karten, Charaktere, Animationen, etc.).

Eine IDE zur Erstellung von Spielen muss es dem/der EntwicklerIn ermöglichen, verschiedene Teile des Spiels zu programmieren, die unterschiedliche Zwecke haben können. Um den unterschiedlichen Anforderungen gerecht zu werden, kann die Umgebung dem/der EntwicklerIn die Verwendung verschiedener Sprachen ermöglichen. Zum Beispiel muss ein/e Online-First-Person-Shooter-EntwicklerIn die 3D-Programmierung intensiv nutzen, aber auch die Kommunikation mit einem Remote-Server über das Netzwerk aufnehmen können. Für diese beiden Anforderungen benötigen


Co-funded by the Erasmus+ Programme of the European Union
The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

wir möglicherweise völlig unterschiedliche Sprachen. Darüber hinaus kann das Spiel Regeln darüber haben, was für den/die SpielerIn erlaubt oder verboten ist, oder die Checkpoints, die der/die Spielerin besuchen muss. Diese Anforderung erfordert in der Regel eine höhere Sprache, in der Regeln leicht geschrieben werden können. Beachten Sie auch, dass das Spiel Bots haben kann, d.h. ComputerspielerInnen, die das Spiel autonom spielen. Wie würden wir die Intelligenz und das Verhalten dieser Bots programmieren?

Viele Umgebungen beinhalten ihre eigene Sprache, die versucht, jeder Anforderung gerecht zu werden, während sie einfach gehalten wird, um die Bedienung der Umgebung zu erleichtern, während andere es ermöglichen, dass allgemeine Sprachen (typischerweise C++) die Leitsprache in der Umgebung sind. Die letzteren schränken die Verwendung anderer Sprachen jedoch nicht ein, da es Werkzeuge und Möglichkeiten gibt, Routinen und Code aus Nicht-C++ in ein C++-Programm einzubetten. Aber auch allgemeine Sprachen wie C++, Rust, Python oder Lua, die in der Spieleentwicklung verwendet werden, werden um eine Reihe von Klassen und Bibliotheksfunktionen erweitert, die fest in die Spiele-Engine integriert sind.

### 1.3 Entwicklungsumgebung und Programmiersprachen – Teil 2

Eine interessante Alternative zur Verwendung der Basis-IDE ist die Verwendung von Spiel-Editoren. Einige Spiele werden mit integrierten Editoren vertrieben. Dies sind die gleichen Werkzeuge, mit denen das Entwicklungsteam die Levels, Karten, Phasenstufen des Spiels erstellt hatte, die auch von den Spieleentwicklern selbst erstellt wurden.

Der Spieler kann den Editor frei verwenden, um sein eigenes Level aufzubauen, einschließlich seiner eigenen Kunst, Bilder und Sounds. Einige Editoren erlauben es, einfache Änderungen an Standardkarten vorzunehmen, während die meisten es erlauben, neue Karten und Levels zu optimieren und zu entwickeln.

Spielgenres, die normalerweise Editoren beinhalten, sind Echtzeitstrategie und First Person Shooters. Aus einem bestimmten Blickwinkel besteht ein editierbares oder modifizierbares Spiel aus den folgenden Elementen:

- Eine Reihe von Ressourcen oder Assets wie Bilder, Sounds, Animationen usw., die den grundlegenden Part der Objekte des Spiels bilden.
- Eine Reihe von Karten, Levels oder Stufen, die eine von vielen Herausforderungen darstellen, die der/die SpielerIn zu meistern hat.
- Eine Software, die die oben genannten Elemente übernimmt und das eigentliche Spiel erstellt, indem sie die Karten und Ressourcen ausführt. Dieses Element wird als "Game Engine" bezeichnet.


Co-funded by the Erasmus+ Programme of the European Union
The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Dieser Ansatz wurde weiterverfolgt. Um einige wirklich komplexe Spiele zu entwickeln, durchlaufen Programmierer zwei Phasen mit unterschiedlichen Ergebnissen. Zuerst bauen sie eine Game Engine, die ein Spiel ausführen kann, aber die Inhalte werden separat erstellt. Die Spiele-Engine ist mit einem speziellen Editor gekoppelt, der verwendet wird, um Levels, Grafiken, Charaktere usw. nur für diese Engine zu generieren. In einem zweiten Schritt vervollständigen sie dann alle Spielinhalte, die die Geschichte des Spiels in Levels aufbauen.

Wenn das Spiel fertig ist, stehen der Öffentlichkeit zwei Produkte zur Verfügung: das eigentliche Spiel mit dem vollen Inhalt zusammen mit der eingebauten Engine und der Editor, der von Enthusiasten verwendet werden kann, die gerne neue Inhalte erstellen. Sie können separat verkauft werden, was dazu führt, dass einige Entwickler die Engine und ihren Editor verwenden, um ein anderes Spiel aufzubauen, nicht nur eine Modifikation oder Erweiterung des ursprünglichen Spiels. Manchmal wird die Game Engine angekündigt und an Dritte oder Personen verkauft, noch bevor ein Spiel mit dieser Engine abgeschlossen wurde. Aber dennoch können diese nicht als "Allzweck"-Spielentwicklungswerkzeuge betrachtet werden, da sie von Anfang an auf eine bestimmte Art von Spiel abzielen. Das ist etwas anderes als einfach "ein Spiel mit eingebautem Editor".
Es gibt eine ständig wachsende Zahl an guten Game Engines.

### 1.4 Entwicklungsumgebung und Programmiersprachen – Teil 3

Abschließend und dem Trend in der Software-Erstellung folgend, bieten Online--Shops wie Valve's Steam, Google Play, Sony PlayStation Store eine Reihe von IDE-Lösungen an, die die Veröffentlichung und Verteilung der Software erleichtern. Die IDE ist in der Lage, das Konto des Entwicklers/der Entwicklerin in jedem dieser digitalen Geschäfte zu nutzen, um ein Spiel ohne Probleme zu veröffentlichen.

Als Fazit dieser Einheit müssen wir die Annehmlichkeit der Verwendung eines generischen Spiel-Entwicklungswerkzeugs bedenken, das sich auf die Einfachheit der Anwendung konzentriert und auf AnfängerInnen ohne Vorkenntnisse in der Programmierung zugeschnitten ist. Und aus der großen Liste der verfügbaren Software ist GameMaker von YoYo Games ein sehr geeignetes Werkzeug für dieses Projekt. Es handelt sich um eine bewährte, verbreitete Lösung mit einer großen Community von AnwenderInnen und mehreren erfolgreichen Produkten.

Co-funded by the Erasmus+ Programme of the European Union
The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

# Kapitel 2 - Game Maker

## 2.1 Merkmale des Game Maker Rahmenwerks. Spieltypen.

"GameMaker Studio" ("GameMaker" in dieser Anleitung) ist eine komplette integrierte Umgebung zur Entwicklung von Spielen. Ursprünglich wurde es so konzipiert, dass AnfängerInnen einfache Spiele ohne große Programmierkenntnisse erstellen konnten. Um dieser Anforderung gerecht zu werden, bot sie mehrere Technologien an, die es ermöglichten, das Äquivalent eines Computerprogramms visuell zusammenzustellen. Allerdings können EntwicklerInnen bei Bedarf Code schreiben und eine Programmiersprache ähnlich C verwenden, aber mit einigen Anleihen aus Javascript. Das Ergebnis ist eine Sprache, die einfach genug ist, um von AnfängerInnen verwendet zu werden.

Die Hauptmerkmale sind:

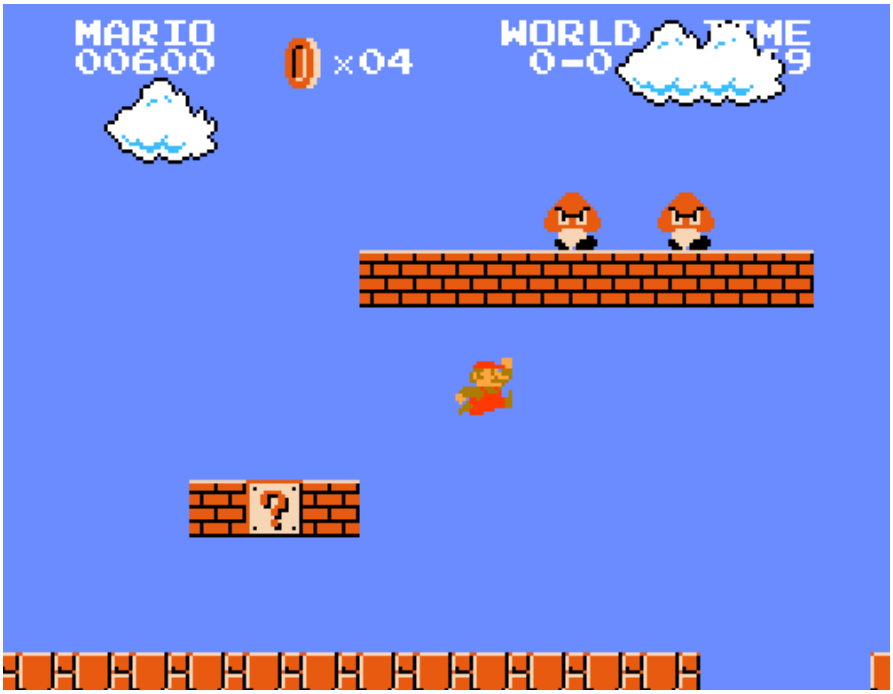
- Integriert. Alle oder die meisten Phasen der Spieleentwicklung können mit integrierten Tools durchgeführt werden. In den meisten Fällen benötigt man keine Unterstützung durch externe Software.
- Es hat eine eigene Sprache (Game Maker Language oder GML), die Programmierern, die mit C, Javascript usw. vertraut sind, nicht unbekannt ist und eng in die Spiele-Engine und Entwicklungsumgebung integriert ist.
- Für einfache Spiele wird kein Code benötigt. Game Maker ermöglicht es dem Entwickler, "DnD" Drag and Drop Programmierung als Alternative zur Verwendung von GML zu verwenden.
- Spiele können für die meisten Betriebssysteme und Spieleplattformen erstellt werden. Und die Integration mit digitalen Vertriebsplattformen ist ebenfalls enthalten.
- Besonders geeignet für verschiedene Arten von Spielen, kann aber mit etwas mehr Aufwand an viele zusätzliche Genres angepasst werden.Integrated. All or most stages of the game development can be done with built-in tools. For the most part, It doesn't need help from external software.

Game Maker zeichnet sich durch die Einfachheit und Nützlichkeit bei der Erstellung von Plattformspielen und Shoot'em up-Spielen aus. Aber auch andere Arten von Spielen können erstellt werden.

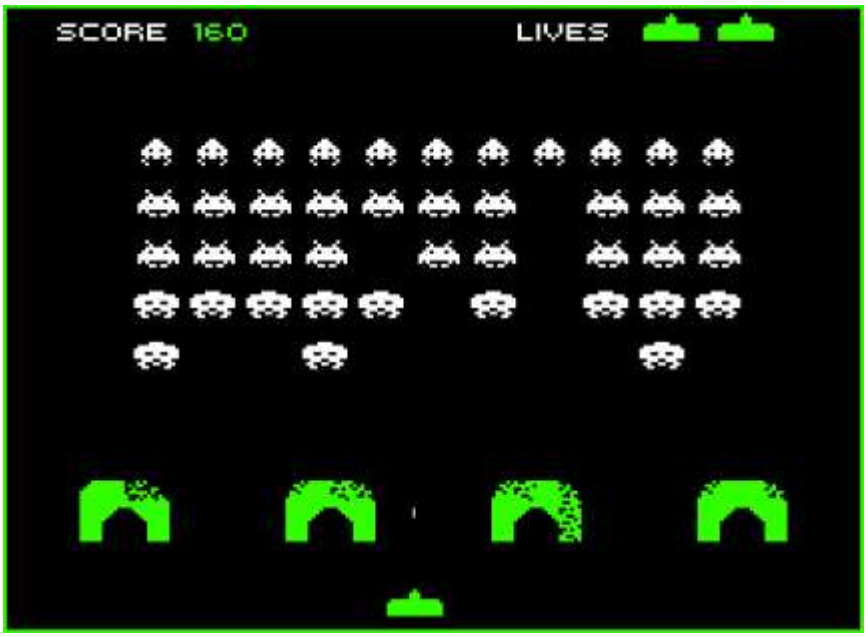
- In Plattformspielen bewegt sich ein Charakter in einer zweidimensionalen Welt, sammelt Objekte, vermeidet Hindernisse und zerstört Feinde. Ein bekanntes Beispiel ist Mario Bros



Co-funded by the Erasmus+ Programme of the European Union
The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.




- Shoot Them up Spiele. In diesem Spiel bewegt sich der Spieler kontinuierlich über eine Karte, entweder fliegend oder gehend, und zusätzlich kann er verschiedene Punkte auf dem Bildschirm erreichen. Feinde erscheinen in der Richtung " vor " dem Spieler, der diese Feinde meiden oder töten muss. Ein ewiges Beispiel ist " Space Invaders "




Co-funded by the Erasmus+ Programme of the European Union
The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

## 2.2 Entwicklungsumgebung

Game Maker ist eine komplette Entwicklungsumgebung. Es enthält alle notwendigen Werkzeuge, um Spiele zu entwickeln und zu testen, ohne dass externe Tools, Geräte oder spezielle Setups erforderlich sind. Die Benutzeroberfläche ist um einen zentralen Rahmen herum aufgebaut, in dem mehrere Arbeitsbereiche zugewiesen werden können. Arbeitsbereiche sind Panels oder Tafeln, auf denen Spielelemente und Ressourcen während der Entwicklung platziert und organisiert werden können, um eine individuellere Arbeitsorganisation zu ermöglichen. Arbeitsbereiche selbst zeigen interne Fenster mit unterschiedlichem Inhalt an, von generischen Objekten mit Eigenschaften über Ereignislisten bis hin zu GML-Code. Jedes Elementfenster, das Teil eines anderen ist oder irgendwie verbunden ist, hat eine Linie, die von einem Element zum anderen gezogen wird.

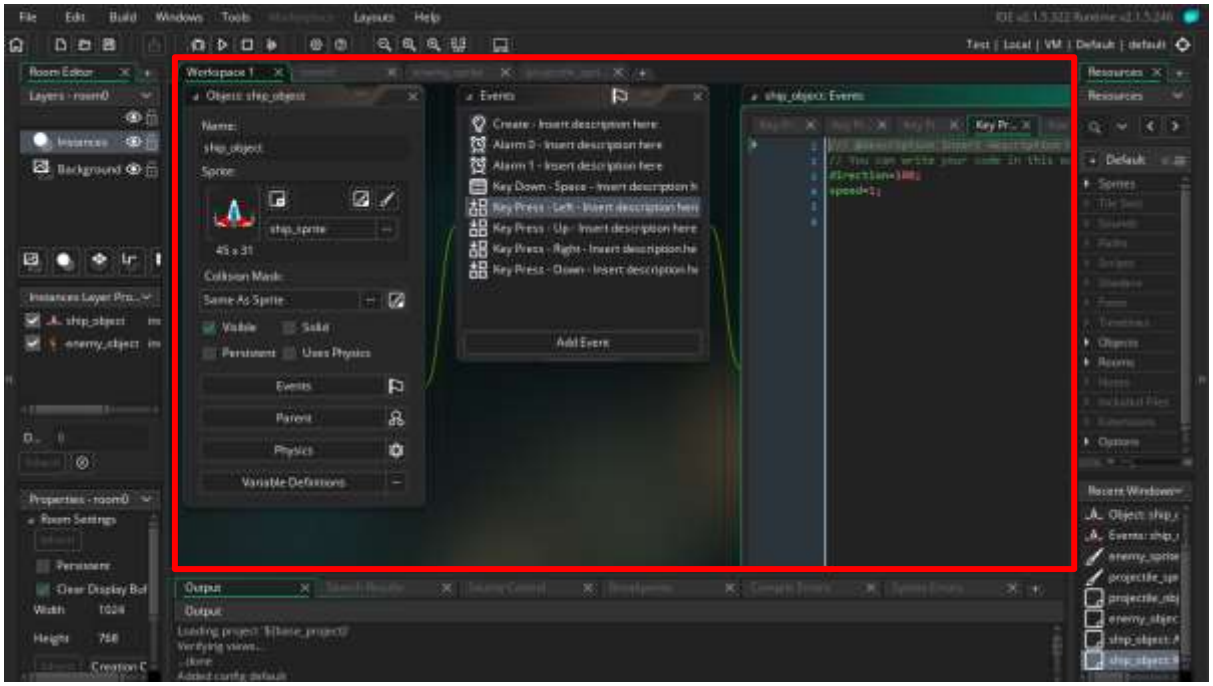
Darüber hinaus gibt es um den zentralen Arbeitsbereich herum Sekundärpaneele. Die meisten von ihnen sind bei Bedarf zusammenklappbar, damit sich der Zentralbereich ausdehnen kann. Diese Panels zeigen:

- Eine Baumstruktur mit allen Ressourcen des Spiels. Dieser Baum ist dauerhaft und ändert sich nur, wenn Ressourcen (oder Anlagen) angelegt oder gelöscht werden,
- Eine Liste von Steuerelementen, um die Eigenschaft einer ausgewählten Ressource zu ändern. Dieses Panel ändert sich, wenn sich der/die EntwicklerIn auf ein anderes Element konzentriert.
- Ein Ausgabefenster, um das Ergebnis der Spielerstellung und -ausführung zu sehen.

Das andere Fenster kann entweder einzeln, über dem Rest oder als Registerkarte im zentralen Bereich erscheinen, wenn ein Spezialwerkzeug gestartet wird. So wird beispielsweise der Bildeditor selten verwendet, aber wenn eine Bildressource bearbeitet wird, wird sie als Tab im zentralen Bereich in den Vordergrund gestellt.



Co-funded by the Erasmus+ Programme of the European Union
The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



Der Game Maker bietet zwei Möglichkeiten, ein Spiel zu erstellen:

- Drag and Drop (oder "DnD"). Dieser Modus ist für AnfängerInnen gedacht, die keine Lust haben oder nicht programmieren wollen und so tun , als würden sie Standard-Spiele bauen. Der Game Maker ermöglicht die Definition der Spiellogik mit Hilfe eines "Visual Scripting Tools". In diesem Modus verschiebt der/die EntwicklerIn Logik-Elemente, die verknüpft und gruppiert sind, um das Verhalten eines jeden Elements festzulegen.
- Game Maker Sprachprojekte. Dieser Modus ist die Standardeinstellung und die Logik des Spiels und des Verhaltens eines jeden Elements wird durch Programmiercode in der GML-Sprache festgelegt.

In diesem Tutorial verwenden wir die Variante GML, da die Komplexität der Beispiele gering sein wird und wir keine komplexe Programmierung durchführen werden, aber wenn wir fertig sind, werden Sie in Zukunft mehr Möglichkeiten haben.

### 2.3 Sprites, Räume, Objekte und Kollisionen.

Spiele zeigen Elemente auf einem Bildschirm in wechselnder und dynamischer Weise an. Was der Spieler sieht, sind einfach "Dinge", die sich innerhalb eines Bereichs oder einer "Welt" bewegen, und oft stürzen, stoßen oder kollidieren Dinge mit anderen Dingen.

Game Maker, aber auch Entwickler und Spieler, vergeben spezifische Namen für verschiedene visuelle Elemente.

Co-funded by the Erasmus+ Programme of the European Union
The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Der Bereich oder die "Welt", in dem das Spiel stattfindet, wird in GameMaker Raum genannt.

Animierte Bilder werden als "Sprites" bezeichnet. Statische oder nicht animierte Bilder sind nur dekorativ und können als Hintergrund bezeichnet werden.

Elemente eines Spiels, die ein Verhalten haben oder das Spiel in irgendeiner Weise beeinflussen, erhalten den Namen "Objekt".

Und schließlich ist "Kollision" das häufigste Ereignis zwischen zwei Objekten, wenn sie sich physisch treffen.

Sprites, rooms, objects and collisions - Ein ausführlicher Leitfaden, der die wichtigsten Elemente eines Spiels behandelt.

## 2.4 Events und Aktionen

Ein Spiel, wenn es läuft oder gespielt wird, ist ein dynamisches System. Es ist etwas, das sich ändert, oft sehr schnell aus SpielerInnensicht. Diese ständige Veränderung im Spiel ist eigentlich das Ergebnis kleiner, sehr häufiger Veränderungen, die als eine größere, komplexere Wendung der Ereignisse wahrgenommen werden. Der Schlüssel, um festzustellen, wie sich das Spiel entwickelt, besteht darin, die grundlegenden Änderungen zu verstehen und wie sie aus ihren Komponenten "Events" und "Aktionen" abgeleitet werden.

Um “Events” und “Aktionen” besser zu verstehen, müssen wir zunächst bedenken, dass ein Computerspiel eigentlich ein Programm ist, das kontinuierlich läuft. Manchmal wird dieses Programm auch als "Game Engine" bezeichnet. Dieses Programm überwacht häufig die Tastatur, Maus und andere Controller; es kann die Zeit ablesen, kennt die Position und Richtung eines jeden Objekts aus dem Spiel, etc. Daher ist die Game Engine die erste, die weiß, dass etwas passiert ist: Entweder wurde eine Taste gedrückt, oder es ist eine Verzögerung verstrichen, oder zwei Objekte haben sich auf dem Bildschirm berührt, etc. Game Maker ruft Ereignisse zu diesen Dingen auf, die auftreten und das Spiel beeinflussen können. Viele Ereignisse treten auf, wenn der Spieler mit dem Computer interagiert. Einige andere Ereignisse treten als Folge der Entwicklung des Spiels auf.

Die Spielmaschine steuert das Spiel und kann den Zustand von allem, was sich darin befindet, zu

Co-funded by the Erasmus+ Programme of the European Union
The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

jedem beliebigen Zeitpunkt ändern. Es kann ein Objekt neu positionieren oder verschieben, ihm Schaden zufügen, ein neues Objekt erstellen, etc. Diese Änderungen erfolgen so, wie es die Programmierung des/der EntwicklerIn vorsieht, wobei die "Game-Engine" nur die Ausführende des Programms ist. Game Maker ruft "Aktionen" zu einer Änderung auf, die vom Entwickler festgelegt werden. Der Schlüssel dazu ist, dass jede Aktion als Ergebnis eines Ereignisses geschieht, auch Aktionen, die zufällig ausgeführt werden, sind mit Ereignissen, mit einer zufälligen Komponente verknüpft,. Das Ziel des Entwicklers ist es, festzulegen, welche “Aktionen” als Folge welcher “Events” stattfinden. Wir werden die “Events” zuerst gründlich analysieren und dann sehen, wie “Aktionen” mit ihnen verknüpft sind.

Es gibt eine lange Liste von vorab festgelegten Ereignissen im Game Maker, aber es steht dem/der SpieleentwicklerIn frei, jede Maßnahme als Reaktion auf ein bestimmtes Ereignis zu ergreifen (bitte beachten Sie die Online-Ressource).

Events and Actions - Ein ausführlicher Leitfaden, der die wichtigsten “Events” in Game Maker behandelt.

### 2.5 Bedingte Aktionen

Manchmal müssen Aktionen unter komplexen Bedingungen durchgeführt werden, für die kein Ereignis zugeordnet ist. In diesen Fällen muss der/die EntwicklerIn eine spezielle Bedingung schreiben, unter der die Aktion ausgeführt wird. Und die Game Engine muss angewiesen werden, den Zustand zu jedem möglichen Zeitpunkt zu überprüfen. Wenn die Bedingung erfüllt ist, wird die Aktion ausgeführt. Dies bietet dem/der EntwicklerIn eine große Flexibilität. Bedingte Aktionen sind jedoch am Ende Aktionen, die mit einem Ereignis verknüpft werden müssen (so funktioniert Game Maker). Wenn die Bedingung jedes Mal überprüft werden muss, dann ist das Ereignis natürlich das "Step"-Ereignis. Und die Aktion ist einfach ein Programm, das zusätzlich die Bedingung prüft.

Conditional Actions - Beachten Sie dieses Dokument, um einen besseren Überblick darüber zu erhalten, wie bedingte Aktionen funktionieren.

### 2.6 Animation und erweiterte Kollisionen

Die Animation in Spielen ist entscheidend für ein gutes In-Game-Erlebnis und für das Eintauchen in das Spiel. Seit es Videospiele gibt, haben ProgrammiererInnen und KünstlerInnen hart daran gearbeitet, das Spiel so realistisch und ansprechend wie möglich zu gestalten. Dazu ist es notwendig, die Bewegungen nachzuahmen, die viele Objekte im realen Leben haben. Zum Beispiel sollte das Fahren eines Autos von oben oder hinten nicht nur dazu führen, dass man das Auto auf einer Straße



Co-funded by the Erasmus+ Programme of the European Union
The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

beobachtet, sondern auch, dass sich die Räder drehen, der Rauch aus dem Auspuff austritt und die Straße die Reifen schüttelt. Es gibt mehrere Techniken, um Animationseffekte in Spielen zu erzeugen, von denen einige gut in den Game Maker integriert sind. Bilder befassen sich auch mit der physischen Darstellung von Objekten, die sich häufig bewegen und mit anderen Objekten kollidieren.

Die am weitesten verbreitete Technik, um Animationen zu erhalten, kommt von der Verwendung einer Reihe ähnlicher Bilder, die zyklisch angezeigt werden. Wenn die Geschwindigkeit, mit der sich die Bilder ändern, stimmt, nimmt der/die SpielerIn die Bilder als Animation wahr. Die schwierige Aufgabe, die bleibt, ist es, eine Reihe von Bildern zu zeichnen, die die bestmögliche Animation erzeugen.

Wir betrachten das “Sprite” als Bild, mit mehreren Teilbildern. Nehmen wir zum Beispiel den folgenden "Streifen" von Bildern. Das “Sprite” würde ein Mädchen nach dem anderen zeigen, das schnell zum nächsten in Folge wechselt, was den Effekt hat, dass das Mädchen tatsächlich geht, indem es seine Beine bewegt (das Bild würde sich zusammen mit der Position ändern).

https://opengameart.org/content/girl-walking-side

Game Maker ist sehr mächtig, wenn es darum geht, “Sprites” zu erstellen und zu verwenden. Es kann eine Datei wie die obige lesen und sie ganz einfach in verschiedene Teilbilder "aufteilen" und so das “Sprite” fast automatisch erstellen. Es kann Bilder verarbeiten, deren Teilbilder in anderen Layouts (z.B. in einer Matrix) ausgerichtet sind.

Die Verwendung von “Sprites” im Game Maker ist ziemlich einfach, sehr leistungsstark und liefert gute Ergebnisse. Aber das Zeichnen oder Erstellen der “Sprites” ist eine andere Geschichte. Um mit der Erstellung von Spielen zu beginnen, gibt es viele kostenlose Spritesets in einigen Websites. Eine dieser Websites ist http://opengameart.org.

“Sprites” legen auch die Objektform fest und je nach Größe des Objekts im Verhältnis zum Raum kann es unverantwortlich sein, die Kollisionen zwischen Objekten zu verwalten, die nur auf den Koordinaten von Objekten basieren. Betrachten Sie zum Beispiel ein kleines Objekt, das sich auf ein großes zubewegt. Meistens muss die Kollisionserkennung verhindern, dass sich Objekte überlappen oder ineinander übergehen. Daher muss das Kollisionssystem nicht nur die Positionen der Objekte, sondern auch die Form und die Abmessungen berücksichtigen.

Co-funded by the Erasmus+ Programme of the European Union
The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Bei der Definition eines „Sprites“ ist es auch nützlich und einfach, die "Kollisionsmaske" des Sprites einzustellen, die für Objekte gilt, die das Sprite anzeigen. Die Kollisionsmaske ist ein Bereich mit einer potentiell beliebigen Form. Typischerweise ist die Kollisionsmaske aus Performance-Gründen ein Rechteck, das bei Bedarf gedreht werden kann, um die Sprite-Form besser anzupassen. Kreise und Ellipsen können auch als Kollisionsmaske eingestellt werden, sind aber langsamer zu berechnen. Schließlich kann in sehr speziellen Fällen die Kollisionsmaske auf die Form aller Teilbilder des Sprites eingestellt werden. Das ist viel, viel langsamer. Beachten Sie, dass Kollisionen 60 Mal pro Sekunde überprüft werden müssen, wenn zwei Objekte sich annähern.

Im Game Maker werden einfache Kollisionen vom/von der EntwicklerIn auf zwei Arten gelöst:

● Mit dem Kollisionsereignis, für das der/die EntwicklerIn den Aktionscode schreiben kann, der sich darum kümmert, was zu tun ist. Dies ist der bevorzugte Weg, wenn Kollisionen für ein Objekt (z.B. ein Raumschiff, das durch den Raum fährt) nicht üblich sind.

● Überprüfung potenzieller Kollisionen bei Aktionen, die mit dem „Step“-Ereignis verbunden sind. Dies ist empfehlenswert, wenn ein Objekt normalerweise mit einem anderen kollidiert. Die Nichtverwendung des Kollisionsereignisses spart einige Zeit, da das Gleiche vom „Step“-Ereignis durch explizite Überprüfung des Codes erreicht werden kann.

Game Maker enthält eine Physik-"Game-Engine", die eine völlig neue Art des Umgangs mit Bewegungen, Kollisionen und physischer Interaktionen zwischen Objekten ermöglicht. Wenn diese Funktion verwendet wird, werden Kollisionen vollständig von der Game Engine verwaltet, was die Entwicklung erheblich vereinfacht. Der Nachteil ist, dass der/die EntwicklerIn für jedes Objekt und jeden Raum physikalische Eigenschaften angeben und wahrscheinlich einige Parameter feinabstimmen muss, um das gewünschte Ergebnis zu erzielen.

Sprites - Website mit Zugriff auf ein Repository von „Sprites“



Co-funded by the Erasmus+ Programme of the European Union
The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

## 2.7 Export und Veröffentlichung von Videospielen

Der Export von Spielen bezieht sich auf die Fähigkeit des Game Maker, ein Programm zu erstellen, das unter einer bestimmten Hard- und Software ausgeführt werden kann, während die Veröffentlichung die weit verbreitete Verfügbarkeit des Spiels ist, das von der breiten Öffentlichkeit erworben werden soll. Traditionell werden beide Aktivitäten von völlig unterschiedlichen Unternehmen durchgeführt. Aber hier beschäftigt sich Game Maker mit beiden Problemen und erleichtert es dem/der EntwicklerIn, seine/ihre Arbeit den SpielerInnen bekannt zu machen.

Das Hauptmerkmal des Game Maker ist die Möglichkeit, auf viele verschiedene Spieleplattformen zu exportieren, ohne etwas am entwickelten Spiel zu ändern. Zu den Zielen (Plattformen oder Systeme, für die SpieleentwicklerInnen ein Spiel erstellen können) gehören Linux (Ubuntu), Mac OS X, Android, iOS, FireTV, Android TV, Windows Desktop, Microsoft UWP, HTML5, PlayStation 4 und Xbox One. Jedes dieser Ziele erfordert spezifische Lizenzen. Game Maker kann Pakete generieren, die auf jedem der oben genannten Systeme installiert werden können. Aber damit das Spiel zu den SpielerInnen gelangt, gibt es danach noch einen langen Weg. Das Spiel muss veröffentlicht und verteilt werden, genau wie Bücher.

In diesem digitalen Zeitalter, in dem alles vernetzt ist, hat das Veröffentlichen von Spielen eine Revolution durchlebt. Spiele wechseln vom physischen Format zu digital herunterladbaren Inhalten. Große Verlage zentralisieren nun die Veröffentlichung dieser digitalen Spiele. Der Weg zur Veröffentlichung eines Spiels wird für EntwicklerInnen verkürzt. Spiele können für SpielerInnen verfügbar sein, sobald ihre Entwicklung abgeschlossen ist. SpieleentwicklerInnen benötigen nur ein Abonnement für die Veröffentlichungsplattform.

Game Maker kann Spiele auf der Steam-Plattform veröffentlichen. Mit Steam können Spiele einige Dienste der Steam-Plattform nutzen, wie z.B. Leaderboards, kostenpflichtige Download-Inhalte, Cloud Storage, etc.

Game Maker bietet verschiedene Lizenzen auf Basis der Zielplattformen sowie das Abonnement oder die Integration auf unterschiedlichen Veröffentlichungsplattformen an.



Co-funded by the Erasmus+ Programme of the European Union

# Kapitel 3 - Praktische Beispiele

## 3.1 – Einführung in die Praxisfälle

In diesem Tutorial erhalten Sie eine Einführung in den Game Maker, indem wir an zwei praktischen Fällen arbeiten. Beide Beispiele sind sehr einfach und erfordern keine Vorkenntnisse in der Programmierung. Sie ergänzen sich gegenseitig und der zweite Ansatz verfolgt einen anderen Ansatz, um fehlende Grundkonzepte aus dem ersten Fall zu betrachten.

## 3.2 – Erstellen Sie ein „Shoot'em up“ Spiel

Der erste praktische Fall hat das Ziel, ein "shoot'em up"-Spiel zu entwickeln, das auf einem vom Spieler gesteuerten Raumschiff basiert. Dieses Schiff kann Projektile abfeuern, um Feinde zu zerstören, die wiederum aus kleineren Raumschiffen bestehen. Feindliche Schiffe erscheinen oben auf dem Bildschirm und bewegen sich nach unten und feuern auf das Raumschiff des Spielers.


Sie können sich auf diese Schritt-für-Schritt-Anleitung beziehen, um den Prozess zu verstehen, der notwendig ist, um ein kleines "Shoot'em up"-Spiel zu erstellen.

Co-funded by the Erasmus+ Programme of the European Union
The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

### 3.3 – Erstellen eines Plattform-Spieles

Das zweite Projekt zielt auf die Entwicklung eines Plattformspiels ab. Darin werden die meisten der gleichen Punkte, die zuerst in der ersten Aktivität angesprochen wurden, überprüft und genauer untersucht. Einige ähnliche Funktionen, die beide Spiele gemeinsam haben, werden mit einem anderen Ansatz gelöst, der dem/der LeserIn eine größere Bandbreite an Möglichkeiten für die Programmierung bietet.

Plattformspiele waren in den letzten Jahren sehr beliebt. In den meisten Fällen sind es lineare Spiele, die nur Geschicklichkeit erfordern, um den Charakter zu bewegen, zu schießen und zu springen um in einer zweidimensionalen Welt weiter zu kommen.

Für diese Aktivität werden wir eine Minimalvariante eines spielbaren Spiels ohne Dekoration aufbauen, die wir dann später hinzufügen werden.

Der/die SpielerIn kann einen Charakter mit nur drei Tasten (Links, Rechts und Leerzeichen – bringt Spielcharakter zum Springen) bewegen.

Sie können sich auf diese Schritt-für-Schritt-Anleitung beziehen, um den Prozess zu verstehen, der notwendig ist, um ein kleines Plattformspiel zu erstellen.



Co-funded by the Erasmus+ Programme of the European Union
The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.