# Map (Spielfeld)

## Eigenschaften des Spielfeldes

Tbd:

Ein leeres Spielfeld erzeugen

Der Map-Wizard

### Vordefinierte Layer-Typen (Ebenen)

#### Die Objects-Ebene

Auf der Object-Layer-Ebene werden spezielle Objekte angelegt, die weder Sprites noch Player-Spielfiguren sind.

Typen des Object-Layers:

|  |  |
| --- | --- |
| Objekt-Typ | BeamPoint |
| Beschreibung | BeamPoints werden als Rechteck-Objekte auf dem Objekt-Layer angelegt.  BeamPoints müssen einen Eindeutigen Namen haben. (Es darf im ganzen Spiel keine BeamPoints mit identischem Namen geben)  Um einen BeamPoint zu benutzen muß ein Sprite mit einem TeleporterBehavior erstellt werden. Berührt der Spieler das Teleporter-Sprite, dann wird der Spieler zu dem BeamPoint teleportiert, die dem TeleporterSprite als BeamPoint-Parameter zugewiesen wurde. |
| Zusatz-Properties | Keine |

|  |  |
| --- | --- |
| Objekt-Typ | MusicPlayer |
| Beschreibung | Abspielen von Hintergrundmusik. Alle Musik-Dateien müssen in folgendem Verzeichnis liegen:  Assets/Sound |
| Zusatz-Properties |  |
| Loop | Abspielen der Playliste in einem Endlos-Loop  Loop=True |
| Start | Nachdem die View geladen ist, soll der Song sofort abgespielt werden:  Start=True |
| Songname | Beispiel  Song1=FunkyMusic.mp3 |

|  |  |
| --- | --- |
| Objekt-Typ | SoundList |
| Beschreibung | Enthält alle Sound-Effekt-Dateien. Jede Sounddatei wird ein Resoucen-Name zugeordnet, über den sie abspielbar ist.  Das System kann .wav und .ogg abspielen.  Alle Sound-Dateien werden als Benutzerdefinierte Eigenschaften angelegt.  Bsp:  Beep=myBeep.wav  Lazer=theamazinglazersound.wav  Alle Sprites haben eine Eigenschaft “Sound”. Wenn man einem Sprite ein Berührungsgeräusch zuordnen möchte, dann schreibt man in die Sound-Eigenschaft den Soundnamen aus der Soundlist.  Bsp:  Sound=Lazer  Alle Sound-Dateien liegen in:  Assets/Sounds |
| Zusatz-Properties | Keine |
| SoundResourceName | Dateiname der Sound-Datei |

#### Die Player-Ebene

TBD

#### Die Sprites-Ebene

TBD

#### Die Map-Ebene

TBD

#### Die BackgroundMap-Ebene

TBD

#### Die Screen-Ebene

TBD

#### Die Image-Ebene

TBD

#### Tilesets (Fliesen-Set)

TBD

##### Spezielle Fliesen

###### Leitern-Fliesen

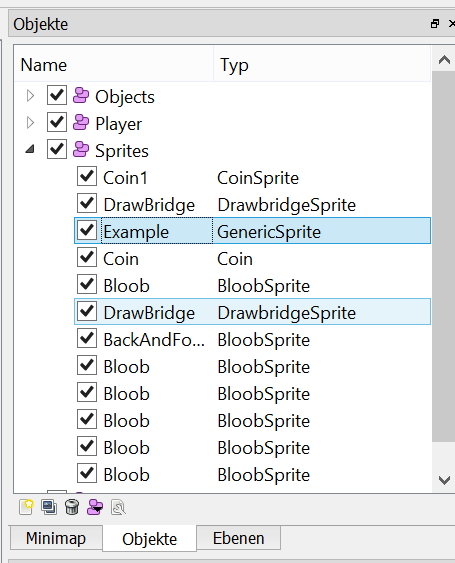
Tbd

###### Space-Fliesen

TBD

# Sprite Properties (Eigenschaften)

Sprite-Eigenschaften können über Tiled direkt als Eigenschaften von Sprite-Objekten zugeordnet werden. Hierzu markiert man das gewünschte Object auf dem Objekt-Layer „Sprites“



|  |  |
| --- | --- |
|  | Alternativ kann man das Sprite auch direkt auf der Map markieren, dazu muss man den Sprite-Objektlayer markiert haben und klickt auf den Befehl „Objekt auswählen“: |
|  | Nun kann man das Sprite mit der Maus direkt auf der Map auswählen: |

|  |  |
| --- | --- |
|  | Anschließend kann man im Eigenschaften-Editor (Property-Editor) zusätzliche Eigenschaften zuordnen:  Durch Klicken auf das Plus-Icon kann man neue Eigenschaften hinzufügen.  In der nachfolgenden Tabelle werden die Eigenschaften und deren Bedeutung aufgelistet. |

## Eigenschaften von Objekten auf dem Sprite-Layer

|  |  |
| --- | --- |
| Eigenschaft | Bedeutung |
| Name | Ein beliebiger Name |
| Typ | Name einer Klasse, die unter Src\Sprites definiert wurde.  Beispiel: GenericSprite, HeartSprite, CoinSprite, etc. Siehe hierzu Erzeugen einer neuen Sprite-Klasse |
| X | X-Position des Sprite auf der Map |
| Y | Y-Position des Sprite auf der Map |

Notiz: Alle anderen Objekt-Eigenschaften werden vom Spiel ignoriert.

## Benutzerdefiniere Eigenschaften von Objekten auf dem Sprite-Layer

|  |  |
| --- | --- |
| Eigenschaft | Bedeutung |
| AssetName | Name des Pfades, wo die Image-Resourcen des Sprite liegen. Beispiel:  AssetName=Coin  In diesem Fall liegen alle Bild-Dateien im Verzeichnis:  Assets\Sprites\Coin |
| Points | Enthält eine Zahl. Wenn der Spieler dieses Sprite berührt, erhält er die hier definierte Punktzahl.  Beispiel:  Points=10  Anmerkung:  Points können auch negativ sein. In diesem Fall würden dem Spieler Punkte abgezogen. |
| Energy | Enthält eine Zahl. Wenn der Spieler dieses Sprite berührt wird ihm Energie hinzugefügt, oder abgezogen. |
| KillSprite | Beispiel:  KillSprite=True  Bei Berührung mit dem Spieler wird das Sprite vom Spielfeld entfernt. (kill)  KillSprite=False  Bei Berührung mit dem Spieler wird das Sprite vom Spielfeld NICHT entfernt. |
| KillPlayer | Beispiel:  KillPlayer=True  Der Spieler stirbt, wenn er dieses Sprite berührt.  KillPlayer=False  Der Spieler stirbt NICHT, falls er das Sprite berührt. |
| Sound | Es wird eine Sound-Datei abgespielt, falls der Spieler dieses Sprite berührt.  Beispiel:  Sound=peng.wav  Alle verfügbaren Sounddateien sind im Verzeichnis:  Assets\Sound |
| Style | Beispiel:  Style=CoinStyle  Die Eigenschaft Style definiert, welcher Style für die Darstellung des Sprites benutzt werden soll.  Für jeden Style muss eine Style-Klasse im Verzeichnis:  Src/SpriteStyles  Definiert sein.  Siehe hierzu Erstellen von Sprite-Styles. |
| Intelligence | Beispiel:  Intelligence=DefaultSpriteIntelligence  Die Eigenschaft definiert, wie sich das Sprite-Objekt auf dem Spielfeld bewegt.  Die Intelligenz-Klasse „DefaultSpriteIntelligence“ definiert ein bewegungsloses Sprite-Objekt.  Alle verfügbaren Intelligenz-Klassen sind im Verzeichnis  Src/SpriteIntelligence  Definiert.  Siehe hierzu: Erstellen von KI´s |
| Behavior | Beispiel:  Behavior=DefaultSpriteBehavior  Die Eigenschaft definiert ein benutzerdefiniertes Kollisionsverhalten, wenn der Spieler dieses Sprite berührt. Durch ein benutzerdefiniertes Behavior kann z.B. implementiert werden, dass der Spieler Teleportiert wird, oder Mauern einstürzen, oder Spielfiguren manipuliert werden.  Behavior-Klassen liegen in:  Src\SpriteBehaviors  Siehe hierzu: Erstellen von Behaviors. |
| Supplies | Beispiel:  Supplies=Nothing  Oder  Supplies=Gun  Die Eigenschaft definiert Ausrüstungs-Items, die der Spieler bekommt.  Supplies-Klassen liegen in:  Src\SpriteSupplies  Siehe hierzu: Erstellen von Supplies |

# Intelligenzklassen (KI´s)

Intelligenzklassen steuern die Bewegungen von Sprite-Objekten auf der Spieloberfläche. Alle verfügbaren Intelligenzklassen müssen im Verzeichnis:

Src\SpriteIntelligence

abgelegt sein.

Momentan gibt es folgende Intelligenz-Klassen:

|  |  |
| --- | --- |
| Klassenname | DefaultSpriteIntelligence |
| Beschreibung | Das Sprite-Objekt bewegt sich nicht und behält eine Position auf der Map. |
| Zusatz-Properties | Keine |

|  |  |
| --- | --- |
| Klassenname | FallDownSpriteIntelligence |
| Beschreibung | Das Sprite-Objekt fällt so lange herunter, bis es auf einer Bodenfliese landet. Danach ist es Bewegungslos. |
| Zusatz-Properties | |
| FallSpeed | Fallgeschwindigkeit, Default 200 |

|  |  |
| --- | --- |
| Klassenname | DropdownAndMoveIntelligence |
| Beschreibung | Das Sprite-Objekt fällt so lange herunter, bis es auf einer Bodenfliese landet. Anschließend geht es in eine Richtung, bis zu einer Mauer oder zu einem Abgrund. Dann geht es in die entgegengesetzte Richtung und so fort. |
| Zusatz-Properties | |
| FallSpeed | Fallgeschwindigkeit, Default 200 |
| WalkSpeed | Gehgeschwindigkeit, Default 100 |

# Style-Klassen

TBD: Was sind Style Klassen

## Verfügbare Style-Klassen

TBD

## Bewegungs-Modes

TBD:

## Benutzerdefinierte Style-Klassen anlegen

TBD

## Den Style-Wizard benutzen

TBD.

# Behavior-Klassen

## Was sind Behavior-Klassen?

TBD

## Verfügbare Behavior-Klassen

Es wurden folgende Behavior-Klassen implementiert:

|  |  |
| --- | --- |
| Klassenname | DefaultSpriteBehavior |
| Beschreibung | Bei einer Kollision passiert nichts. |
| Zusatz-Properties | |
| Keine! |  |

|  |  |
| --- | --- |
| Klassenname | TeleporterBehavior |
| Beschreibung | Bei einer Berührung mit dem Spieler wird der Spieler an einen anderen Ort teleportiert. Über die Properties muß definiert werden, wohin der Spieler gesetzt wird. |
| Zusatz-Properties | |
| ViewName | Name der View, in die gewechselt werden soll. |
| BeamPoint | Name des BeamPoint, zu dem der Spieler teleportiert werden soll. |

Anmerkung: Man kann die Properties BeamPoint oder ViewName verwenden, nicht beide aufeinmal.

Anmerkung: BeamPoints werden auf dem Objekt-Layer angelegt. Siehe auch unter BeamPoints.

## Benutzerdifinierte Behavior-Klassen anlegen

TBD

## Den Behavior-Wizard benutzen

TBD