A picture containing clipart, vector graphics

Description automatically generatedIcon

Description automatically generated**A picture containing text

Description automatically generatedGraphical user interface

Description automatically generated with medium confidenceUne image contenant texte, signe, extérieur

Description générée automatiquement**

Workshop 3 | Pixelohrhasen

xx. xxxxxxxxxxx 2022

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Pixelohrhasen**

In dieser Aufgabenserie lernst du ein Spiel mit Pygame Zero zu erstellen. Du kannst die Aufgaben allein oder zusammen mit deiner Nachbarin lösen. Löse die Aufgaben der Reihe nach und versuche sie aufmerksam durchzulesen und zu verstehen. **Falls du etwas nicht verstehst oder bei einer Aufgabe stecken bleibst, kannst du jederzeit eine Tutorin um Hilfe fragen.**

# Mu Editor

Um in Python programmieren zu können, brauchst du einen **Editor** oder eine Entwicklungsumgebung, in der du **deinen Code schreiben** kannst. Im Workshop verwenden wir den **Mu Editor**. Falls du zuhause gerne weiter coden möchtest, kannst du Mu auch ganz einfach auf deinem Computer installieren (frage vorher deine Eltern um Erlaubnis). Den Download für den Mu Editor findest du hier: <https://codewith.mu/en/download>

Mu ist kostenlos und Python 3 sowie Pygame Zero sind bereits vorinstalliert.

# Den Mu Editor verwenden

A picture containing vector graphics

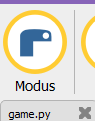
Description automatically generatedWenn du Mu zum ersten Mal startest, kannst du auswählen, welche Art von Programm du schreiben möchtest. Für ein normales Python Programm wähle Python 3. Da wir im Workshop ein Spiel mit **Pygame Zero** entwickeln wollen, wähle Pygame Zero aus.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

A cartoon of a person

Description automatically generated with medium confidenceDu kannst diese Einstellung jederzeit ändern, indem du oben links auf "Modus" klickst.



Jetzt kannst du anfangen deinen Code zu schreiben.

Alles was mit einem **"#"** beginnt ist ein **Kommentar** und gehört **nicht** zum Code. Du kannst diese Zeile löschen oder ignorieren. Kommentare können nützlich sein, um dich später daran zu erinnern, wozu ein gewisser Teil des Codes dient. Du kannst jederzeit eigene Kommentare schreiben, indem du eine Zeile mit "#" beginnst.

Shape

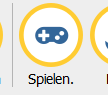
Description automatically generated with low confidenceShape

Description automatically generated with low confidenceEin Bild, das Text enthält.

Automatisch generierte Beschreibung

# Ein Spiel mit Pygame Zero kreieren

## Aufgabe 1: Das Spielfenster

Da du nun den Mu Editor etwas kennengelernt hast, können wir anfangen ein Spiel zu programmieren. Als erstes brauchen wir ein **Fenster**, in dem das Spiel später erscheinen wird. Dafür müssen wir die Höhe (Englisch height) und die Breite (Englisch width) des Fensters angeben. Kopiere den folgenden Code in deinen Mu Editor und klicke auf “Spielen”, um den Code auszuführen.

Icon

Description automatically generated WIDTH = 600

HEIGHT = 600

Mit "Stopp" kannst du das Programm wieder beenden.

Icon

Description automatically generated

Probiere nun die beiden Zahlen zu verändern und schaue was passiert.

**WIDTH** und **HEIGHT** sind sogenannte Konstanten. Sie bleiben während der ganzen Zeit gleich, in der das Programm ausgeführt wird. Namen von Konstanten werden in Python immer in Grossbuchstaben geschrieben.

Icon

Description automatically generatedFür unser Spiel werden wir eine **Fenstergösse von 700 x 700** **Pixel** verwenden, setzte WIDTH und HEIGHT also beide auf 700.

## Aufgabe 2: Der Hase, unsere Spielfigur

Icon

Description automatically generatedDa wir jetzt ein Fenster haben, wollen wir unsere Spielfigur, den Hasen, einfügen. Dazu musst du den folgenden Code hinzufügen:

hase = Actor('bunny1\_stand')

hase.x = 350

hase.y = 350

Icon

Description automatically generatedIn der 1. Zeile dieses Codes erstellen wir unsere **Spielfigur**. Diese ist ein **„Actor“** (von Englisch „action“, also jemand der etwas tut). In Klammern geben wir den **Namen des Bildes** an, welchen wir verwenden wollen. Dabei ist es wichtig, dass sich das Bild **im images Ordner** befindet und der Name in Anführungszeichen geschrieben wird.

Wir geben unserer Spielfigur den Namen **„hase“** damit wir sie später wieder ansprechen können.

Mit **hase.x** und **hase.y** kannst du die **Position** der Figur im Spielfenster anpassen. Wir wollen den Hasen in der Mitte platzieren. Da unser Spielfenster 700 x 700 Pixel gross ist, können wir hase.x und hase.y beide auf 350 setzen (die Hälfte von 700).

Kopiere jetzt noch den folgenden Code ganz ans Ende deines Programms:

def draw():

screen.clear()

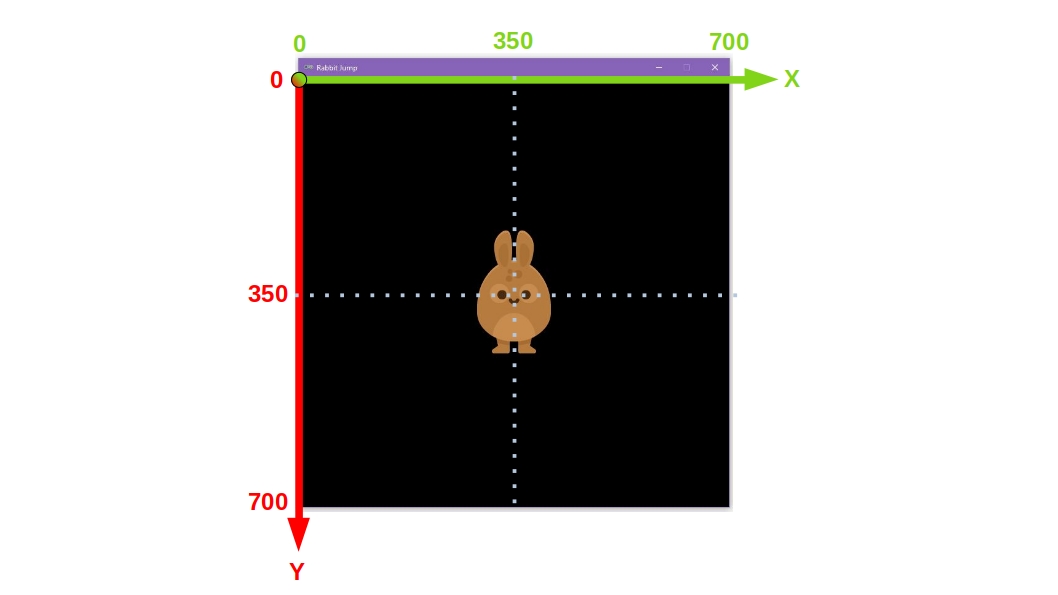
hase.draw()

Das ist eine Funktion, die dafür verantwortlich ist, etwas im Fenster zu zeichnen („draw“ heisst auf Englisch zeichnen). Zum Thema Funktionen lernst du mehr in der nächsten Aufgabe.

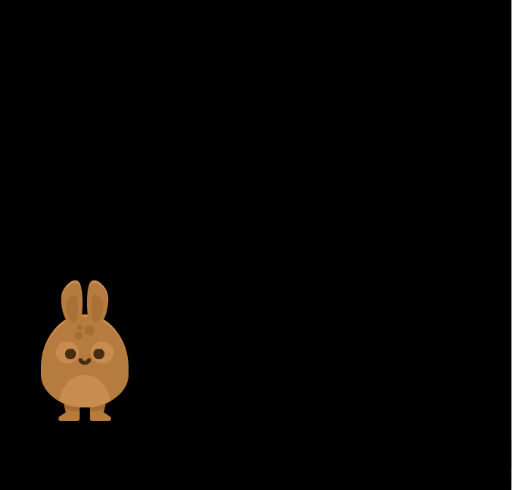
Icon

Description automatically generatedJetzt kannst du auf „Spielen“ klicken und der Hase sollte in der Mitte des Bildschirms erscheinen. Klicke auf „Stopp“ und schaue was passiert, wenn du die Zahlen von hase.x und hase.y veränderst.

Schau dir das folgende Bild an, um zu verstehen, wie Dinge im Fenster platziert werden können:

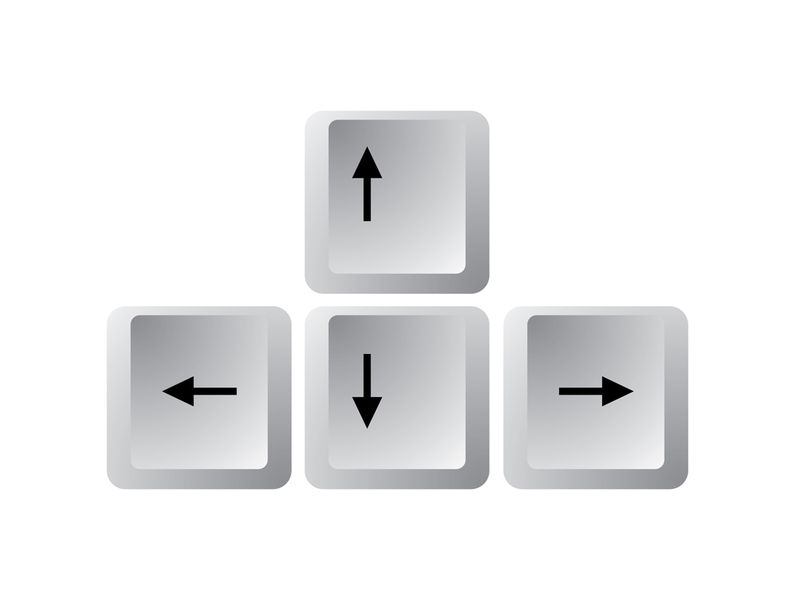


Versuche jetzt den Hasen ungefähr so in der unteren linken Ecke zu platzieren:



## Aufgabe 3: Den Hasen bewegen

Damit sich unser Hase bewegen kann, benutzen wir **die „update“ Funktion**. Eine Funktion ist ein Teil eines Programms oder ein **Unterprogramm**. Funktionen helfen uns ein grosses Programm **aufzuteilen** und übersichtlich zu machen.

Als erstes wollen wir den Hasen nach links und rechts laufen lassen, wenn wir die **Pfeiltasten** für links oder rechts drücken:

Logo

Description automatically generated

In Pygame Zero (und anderen Game Engines) gibt es für **jede Taste auf der Tastatur oder Maus einen Namen**. Die linke Pfeiltaste heisst LEFT und die rechte Pfeiltaste heisst RIGHT. Mit **keys.LEFT** und **keys.RIGHT** können wir die Tasten ansprechen (Keys heisst auf Englisch Tasten).

Füge den folgenden Code ganz am Ende ein:

def update():

if keyboard[keys.RIGHT]:

hase.x = hase.x + 1

if keyboard[keys.LEFT]:

hase.x = hase.x - 1

In diesem Code-Stück lernst du ein weiteres wichtiges Element des Programmierens kennen, das **if-statement**. Das wort **“if”** bedeutet auf Englisch so viel wie **“wenn”** oder **“falls”**. Damit wollen wir signalisieren, dass wir etwas tun wollen, **nur falls etwas Bestimmtes zutrifft**. Schaue dir den folgenden Code-Ausschnitt an:

if keyboard[keys.RIGHT]:

hase.x = hase.x + 1

**Icon

Description automatically generatedKannst du dir vorstellen was hier passiert?**

**Tipp:** Keyboard heisst auf Englisch Tastatur (falls du nicht mehr weisst, was hase.x bedeutet, schau nochmals bei der letzten Aufgabe nach)

Hast du herausgefunden, was dieser Code tut? Genau, der Hase **bewegt sich nach rechts** (hase.x + 1), immer wenn die rechte Pfeiltaste gedrückt wird. Der zweite Teil des Codes tut das Gleiche für die linke Seite (hase.x - 1), wenn die linke Pfeiltaste gedrückt wird.

Klicke jetzt auf “Spielen” und schau, ob du den Hasen bewegen kannst.

**Icon

Description automatically generated**Der Hase bewegt sich noch etwas langsam. **Versuche jetzt herauszufinden, wie du ihn schneller machen kannst.**

**Tipp:** Du musst nur 2 bestimmte Zahlen anpassen, um die Geschwindigkeit des Hasen zu ändern.

## Aufgabe 4: Ein hüpfender Hase

## 

Der Hase kann jetzt nach links und rechts gehen, aber natürlich möchte er auch hüpfen können, wie Hasen das eben machen. Wir wollen den Hasen hüpfen lassen, wenn wir die Leertaste (auch Space genannt) drücken.

Wenn der Hase springt, muss er danach natürlich auch wieder zurück auf dem Boden landen.

Füge zunächst den folgenden Code ganz unten zu deiner „update“ Funktion hinzu:

if hase.y < 500:

Icon

Description automatically generated hase.y = hase.y + 2

Dieser Code sorgt dafür, dass der Hase wieder **zurück auf den Boden** kommt, wenn hase.y kleiner als 500 ist (also weiter oben als die Startposition).

Natürlich kann der Hase auch nicht unendlich hoch springen. Um das zu verhindern kannst du einen Boolean nutzen. Ein **Boolean ist eine spezielle Art** **von Variablen** die nur **zwei Zustände** haben kann, **True** (wahr) oder **False** (falsch). Da wir diese Variable immer wieder brauchen werden, ist es wichtig diese Zeile ganz oben im Code (z.B. nach WIDTH und HEIGHT), ausserhalb der Funktionen hinzuzufügen:

jump\_down = False

Jetzt kannst du die “update“ Funktion ergänzen:

if keyboard[keys.SPACE]:

global jump\_down

if jump\_down and hase.y < 500:

return

elif jump\_down and hase.y >= 500:

jump\_down = False

elif hase.y >= 200 and not jump\_down:

hase.y = hase.y - 5

elif hase.y < 200:

jump\_down = True

Icon

Description automatically generatedHier wird wieder ein if-statement gebraucht. Mit **„elif“** können wir ein if-statement ergänzen, wenn aus mehreren möglichen Situationen prüfen möchten, ob eine zutrifft.

**Starte das Spiel und versuche den Hasen hüpfen zu lassen. Fällt dir etwas auf?**

Genau! Jetzt müssen wir noch dafür sorgen, dass der Hase nicht in der Luft bleibt, solange wir die Space Taste immer wieder drücken und aus der Luft nach oben springt.

Du kannst die folgende Funktion nach der „update“ Funktion im Code hinzufügen:

def on\_key\_up(key):

if key == keys.SPACE:

global jump\_down

Icon

Description automatically generated jump\_down = True

**Schaue dir die Funktion noch etwas genauer an kannst du dir vorstellen was on\_key\_up bedeutet und was die Funktion in etwa macht?**

Super! Jetzt kann der Hase endlich hüpfen!

**Aufgabe 5: Eine schöne Landschaft**

Da ein schwarzer Hintergrund nicht sehr spannend ist, kannst du ganz einfach ein Bild als Hintergrund einfügen.

Füge dazu einfach diese Codezeile irgendwo in der „draw“ Funktion ein:

screen.blit('background', (0, 0))

Diese fügt im Fenster das Bild mit dem Namen „background“ als Hintergrund ein.

Wenn du jetzt das Spiel startest, steht der Hase in einer Landschaft.

**Aufgabe 6: Karotten und Monster**

Damit unser Hase auch etwas zu tun hat, wollen wir Karotten hinzufügen, die durch die Luft fliegen und die der Hase einsammeln muss.

Als erstes brachen wir einen Ort, um die ganzen Karotten zu speichern. Dafür können wir eine Liste verwenden. Eine Liste in Python verhält sich genau wie eine Liste im Alltag (z.B. Einkaufsliste). Darin kannst du mehrere Dinge speichern, hinzufügen oder wider löschen.

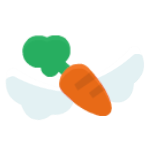
In Python kannst du so eine Liste erstellen:

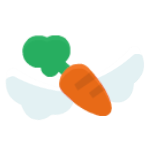
carrots = []

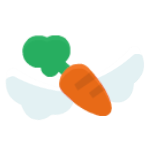
Diese Liste heisst carrots (also Karotten auf Englisch) und ist noch leer. Füge diese Zeile weiter oben in deinem Code hinzu, z.B. nach dem erstellen des Hasen.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Jetzt können wir eine Funktion erstellen, mit der die Karotten erstellt und in die Liste eingefügt werden.

Diesen Codeausschnitt kannst du ganz unten in deinem code hinzufügen:

def make\_carrots():

if random.randint(0,100) < 1:

y = random.randint(70, 350)

carrots.append(Actor('carrot\_fly', (0, y)))

for carrot in carrots:

carrot.x = carrot.x + 3

if carrot.x >= 750:

carrots.remove(carrot)

Icon

Description automatically generatedDas sieht auf den ersten Blick sehr kompliziert aus, aber keine Angst wir schauen uns alles genau an um zu verstehen was hier passiert.

Die Zeile **def make\_carrots():** ist der **Name** der Funktion mit diesem können wir sie später an einem andern Ort im Code aufrufen, wenn wir sie brauchen.

Mit diesem Codeabschnitt fügen wir der carrots Liste zufällig Karotten hinzu:

if random.randint(0,100) < 1:

y = random.randint(70, 350)

carrots.append(Actor('carrot\_fly', (0, y)))

Das wort “random” bedeuted auf Englisch zufällig. In Python können wir z.B. **random.randint(1,10)** verwenden um eine **zufällige Zahl zwischen 1 und 10** zu bekommen.

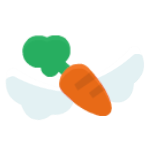
Icon

Description automatically generatedIm Code wird random gleich zweimal verwendet. Beim ersten Mal wählen wir eine zufällige Zahl zwischen 0 und 100. Diese zahl ist die Wahrscheinlichkeit das eine Karotte erscheint. Da diese Funktion vom Programm sehr häufig aufgerufen wird müssen wir aufpassen, dass wir nicht zu viele Karotten haben. Deshalb erstellen wir nur dann eine Karotte, wenn die zufällige Zahl genau 0 (also < 1) ist.

Schaue dir die Zeile **y = random.randint(70, 350)** an. **Hast du schon eine Idee was diese macht?**

Genau! Hier wird zufällig die y-Position (die Höhe) der Karotte bestimmt an der sie erscheinen wird.

Mit **carrots.append(Actor('carrot\_fly', (0, y)))** fügen wir der carrots Liste eine Neue Karotte hinzu. Die Karotte ist genau wie der Hase ein **„Actor“** und hat eineAnfangsposition (0, y). Mit. append können wir etwas in eine Liste einfügen.

Als nächsten kommt ein besonders interessanter Teil des Codes:

for carrot in carrots:

carrot.x = carrot.x + 3

if carrot.x >= 750:

carrots.remove(carrot)

Icon

Description automatically generatedDas ist ein **„for-loop“** (for-Schleife) und wird sehr häufig beim Programmieren verwendet. Einen for-loop können wir verwenden, wenn wir mehrere Dinge (hier Karotten) haben, mit denen wir das gleiche tun wollen (in unserem Fall nach rechts fliegen lassen). Mit **for carrot in carrots:** wählen wir alle Karotten in der carrots Liste aus. **Mit carrot.x = carrot.x + 3** verschieben wir jede einzelne Karotte um 3 nach rechts.

Schliesslich prüfen wir noch ob die Karotte schon wieder als 750 nach rechts gegangen ist, also schon aus dem Bild ist. Wenn dies der fall ist, brauchen wir die Karotte nicht mehr und können sie mit **carrots.remove(carrot)** aus der carrots Liste **löschen**.

So weit so gut!

Füge jetzt noch diesen Codeabschnitt in die **draw Funktion** ein, um alle Karotten aus der carrot Liste auf dem Bildschirm darzustellen:

for carrot in carrots:

carrot.draw()

Als letztes müssen wir die **make\_carrots** Funktion noch in der **update Funktion** aufrufen, damit laufend neue Karotten generiert werden. Füge diese Zeile am Anfang der upadte Funktion ein:

make\_carrots()

Wenn du jetzt das Spiel startest, solltest du fliegende Karotten sehen.

Um das Spiel noch etwas schwieriger zu machen, fügen wir noch ein paar Stachelige Monster hinzu.

Diese funktionieren genau wie die fliegenden Karotten. **Kannst du dir vorstellen was du tun musst, um die Monster hinzuzufügen?**

Die Lösung findest du auf der nächsten Seite.

Erstelle eine Liste für die Monster:

monsters = []

Füge eine neue Funktion hinzu:

def make\_monsters():



if random.randint(0, 300) < 1:

y = random.randint(70, 350)

monsters.append(Actor('monster1', (0, y)))

for monster in monsters:

monster.x = monster.x + 3

if monster.x >= 750:

monsters.remove(monster)

Passe die draw Funktion an:

for monster in monsters:

monster.draw()

und die update Funktion:

make\_monsters()

**Icon

Description automatically generatedWenn du dir nicht sicher bist, was wo hinkommt, schaue dir die komplette Lösung an oder frag eine Tutorin.**

**Aufgabe 7: Punkte und Game Over**

Als letztes wollen wir noch Punkte und Leben hinzufügen. Punkte können verdient werden, wenn der Hase eine Karotte „frisst“, also berührt. Berührt er jedoch ein Monster, soll er ein Leben verlieren. Sind alle Leben verbraucht, endet das Spiel.

Dafür definieren wir als erstes die Anzahl Punkte (score) und Leben (lifes) im oberen Teil des codes z.B. nach den Listen für die Karotten und Monster:

score = 0

lifes = 3

Wir starten mit 0 Punkten und 3 Leben.

Damit die Punkte und die Icons dazu im Spiel angezeigt werden, füge den folgenden Code in die draw Funktion ein:

screen.blit('carrot\_count', (20,640))

screen.blit('lifes', (150, 635))

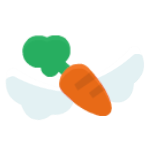
screen.draw.text(f"{score}", (80, 640), color='black', fontsize=60)

screen.draw.text(f"{lifes}", (210, 640), color='black', fontsize=60)

Als nächstes müssen wir überprüfen, **ob der Hase eine Karotte oder ein Monster berührt**. Das können wir mit dem Folgenden Code in der **update Funktion** machen:

global score

global lifes

 for carrot in carrots:

if hase.colliderect(carrot):

score += 1

carrot.x = 750

for monster in monsters:

 if hase.colliderect(monster):

lifes -= 1

monster.x = 750

if lifes == 0:

exit()

Icon

Description automatically generated

**Schaue dir den Code an und überlege dir in welcher Zeile überprüft wird ob der Hase eine Karotte oder ein Monster berührt.**

Genau! Mit **hase.colliderect(carrot)** weden Brührungen mit Karotten geprüft und mit **hase.colliderect(monster)** mit Monstern.

Icon

Description automatically generated**Spiele jetzt das Spiel und schau was passiert, wenn der Hase keine Leben mehr hat. Kannst du den Ausschnitt im Code finden, der das Spiel beendet?**

**Super! Wir haben ein funktionierendes Spiel!** Wenn du etwas noch nicht ganz verstanden hast, kannst du jederzeit eine Tutorin fragen oder dir die Lösung anschauen.

Shape

Description automatically generated with low confidenceDu kannst jetzt die Zusatzaufgaben lösen. Wenn du noch weiter Coden möchtest schau dir die Zusatzaufgaben 1 und 2 an oder entwirf ein eigenes Storyboard bei Aufgabe 3.

Shape

Description automatically generated with low confidence

**Zusatzaufgabe 1: Ein neuer Hase**

****Neben dem braunen Hasen gibt es auch noch einen violetten. Du findest das Bild im Ordner „images“. **Kannst du anstatt den braunen den violetten Hasen ins Spiel einbauen?**

**Zusatzaufgabe 2: Animationen**

Damit das Spiel realistischer wirkt, wäre es toll, wenn der Hase beim hüpfen seine Beine hebt. Im „images“ Ordner gibt es das entsprechende Bild dazu. Wir können das aktuelle Bild des Hasen im Spiel ganz leicht ändern. Du musst nur den folgenden Code in die update Funktion einfügen:

if hase.y < 500:

hase.image = "bunny1\_jump"

else:

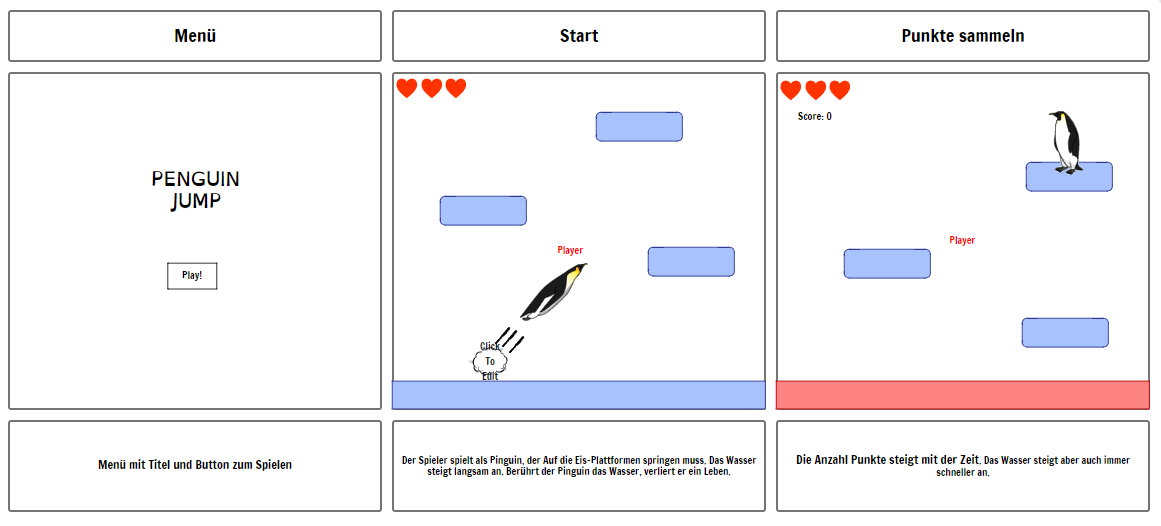
hase.image = "bunny1\_stand"

**Schaue dir die Namen der Bilder im „images“ Ordner und überlege dir was hier wohl passiert. Probiere aus, ob du das gleiche sowohl mit dem violetten und dem braunen Hasen machen kannst.**

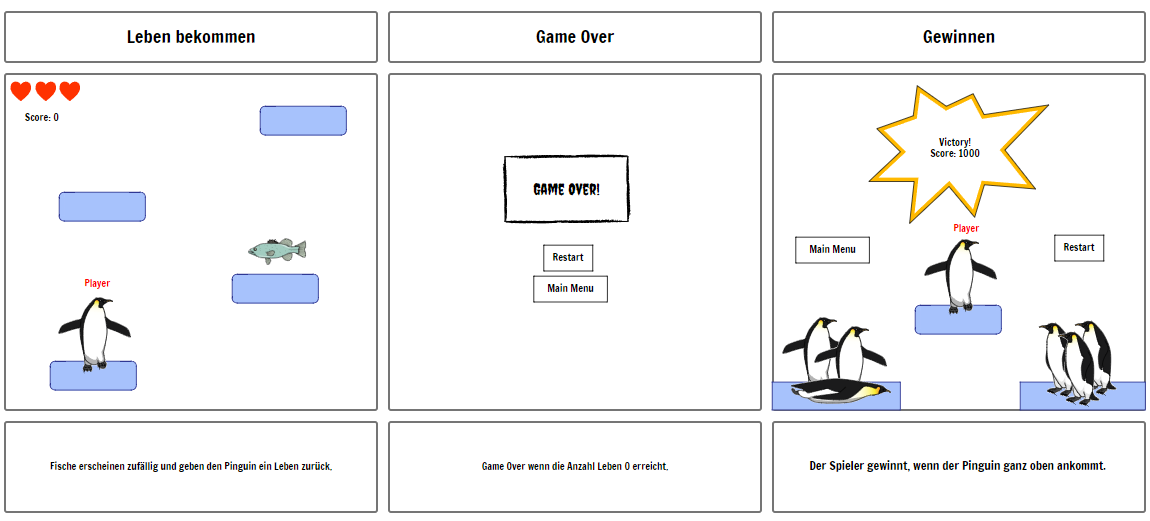
Logo

Description automatically generated**Zusatzaufgabe 3: Kreiere ein Storyboard**

Werde jetzt selbst kreativ! Game Developer benutzen oft Storyboards, um Ideen für Spiele darzustellen. Ein Storyboard könnte zum Beispiel so aussehen:



A cartoon of a person

Description automatically generated with medium confidence

Eine Vorlage für ein Storyboard findest du auf der nächsten Seite.

Icon

Description automatically generated

**Lösungen und Unterlagen**

Alle Lösungen und Unterlagen zum Workshop findest du auf Dropbox: <https://www.dropbox.com/sh/trq796ttod52yxv/AAC1RtvkKB8GgxaBlr4bQWa_a?dl=0>

A picture containing clipart, vector graphics

Description automatically generated

A picture containing text

Description automatically generated