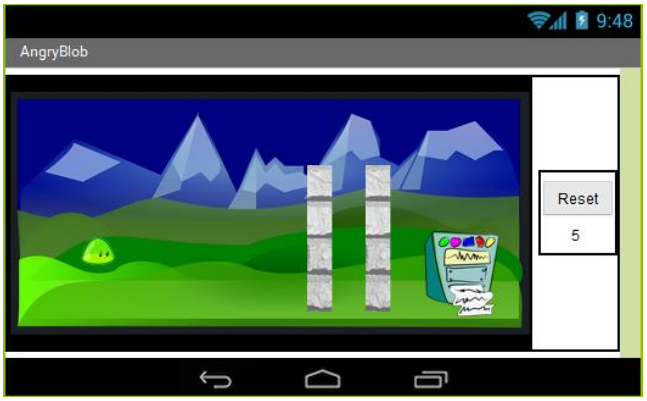
Angry Blob

Ihr habt euch also für **AngryBlob** entschieden. Dies ist ein lustiges Spiel, bei dem es darum geht, den **Blob** zu werfen um den Super puter zu zerstören. Dieses Arbeitsblatt wird euch dabei helfen eine App zu erstellen, die..

... einen auf einem **Spielfeld** erscheinen lässt.

... es dem **Spieler** ermöglicht den durch **Wischen** zu werfen und

... die Verteidigung des Supercomputers bzw. den Supercomputer selbst zerstört wenn er getroffen wird.



Schwierigkeitsgrad: ★★★★

Anlegen des Projekts

Wie bei jeder neuen App müsst ihr zunächst ein neues Projekt anlegen.

1.) Erstellt jetzt euer Projekt und gebt ihm einen ansprechenden Namen

2.) Verbindet den App Inventor wieder wie auf dem ersten Arbeitsblatt beschrieben mit dem Handy.

CC BY SA

Angry Blob

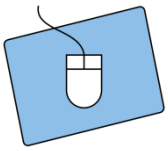
Der Aufbau eurer App

Als nächstes müsst ihr euch Gedanken darüber machen, wie eure App aussehen soll. Eure App braucht Platz für folgende Dinge (die ihr der Reihe nach einbauen werdet):

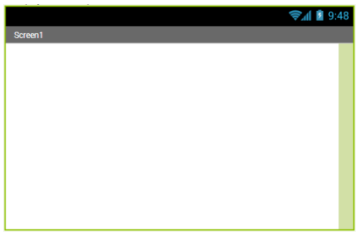
- Eine ***Spielfläche,*** auf der sich der sich euer Blob bewegen kann,
- einen ***Reset-Knopf,*** mit dem ihr das Spiel neu anfangen könnt und
- eine ***Lebensanzeige,*** die nach jedem Versuch ein Leben runterzählt.

Das sind schon recht viele Sachen. Damit diese alle Platz haben, sollt ihr zunächst die Ausrichtung eures *Screens* von ***Unspecified*** in ***Landscape*** ändern:

Mit ***Landscape*** ist bei Apps eine bestimmte Ausrichtung des Bildschirms gemeint, nämlich die horizontale Ausrichtung. Möchte man eine vertikale Ausrichtung haben, dann wählt man ***Portait***.

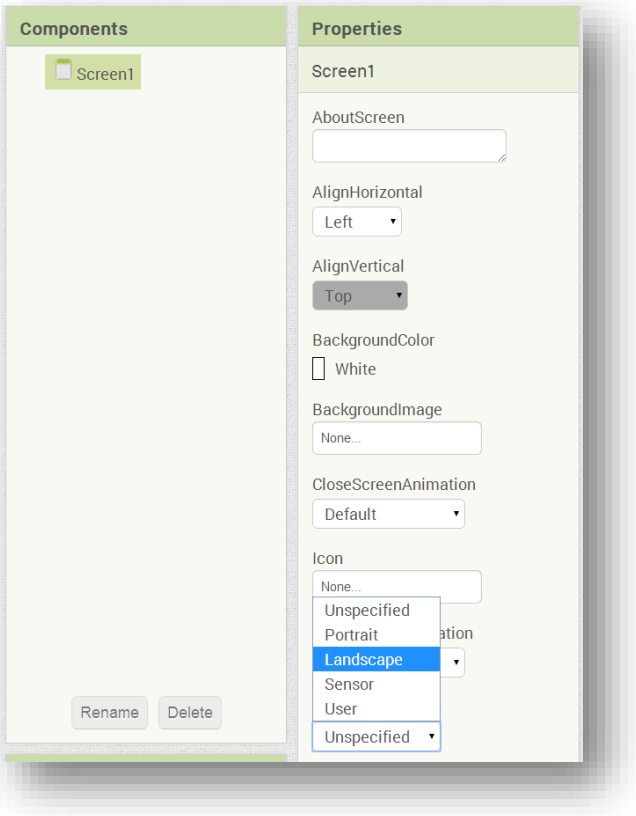


1.) Wählt euren *Screen1* unter **Components** aus.
2.) Sucht unter **Properties** den Eintrag *ScreenOrientation.*
3.) Ändert den Wert von *Unspecified* in *Landscape.*



Landscape

Portrait



CC BY SA

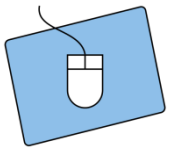
Die Bestandteile eurer App

Nun könnt ihr anfangen, die einzelnen Elemente eurer App hinzuzufügen und diese mit Hilfe der vorgestellten ***Screen Arrangements*** sinnvoll anzuordnen.

- Für die ***Spielfläche*** braucht ihr ein ***Canvas***.
  - Das Canvas findet ihr in der *Palette* im Unterpunkt *Drawing and Animation*.

Ein Canvas ist eine Leinwand auf der sich Objekte wie euer Blob bewegen können.

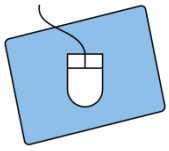
Für den ***Reset-Knopf*** braucht ihr einen ***Button***, mit dem Text Reset".
Und für die ***Lebensanzeige*** braucht ihr ein ***Label*** mit dem Text „5" bzw. der Anzahl an Leben, die euer Blob haben soll



1.) Macht euch Gedanken darüber, wie ihr die Elemente positionieren wollt.
2.) Zieht die ScreenArrangements, die ihr für euer Layout braucht, hinein.
3.) Fügt die Bestandteile (Canvas, Button und Label) eurer App hinzu.

Der erste Test

Nachdem ihr die Bestandteile sinnvoll angeordnet habt, könnt ihr direkt einmal testen, wie das Ganze auf eurem Smartphone aussieht.



1.) Startet dazu die App auf dem Smartphone wenn ihr dies noch nicht gemacht habt.
2.) Gefällt euch die Anordnung? Wenn ja, könnt ihr weitermachen, ansonsten überarbeitet sie einfach nochmal.

Lasst die App einfach auf dem Smartphone laufen. Der App Inventor aktualisiert alle Änderungen, die ihr vornehmt, und zeigt sie euch direkt an, ohne dass ihr die App neu starten müsst.

CC BY SA

Umbenennen nicht vergessen ;)

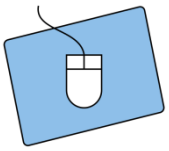
Da ihr jetzt die ersten Bestandteile eurer App fertig habt, ist es an der Zeit diese wieder mit sinnvollen Namen zu versehen, damit ihr sie im Blocks-Editor besser unterscheiden könnt.

Ein Hintergrund für eure Spielfläche

Damit sich der Blob auf der Spielfläche wohl fühlt, soll diese einen Hintergrund bekommen. Ladet dazu einen (von zwei möglichen) Hintergründen aus folgendem Ordner hoch:

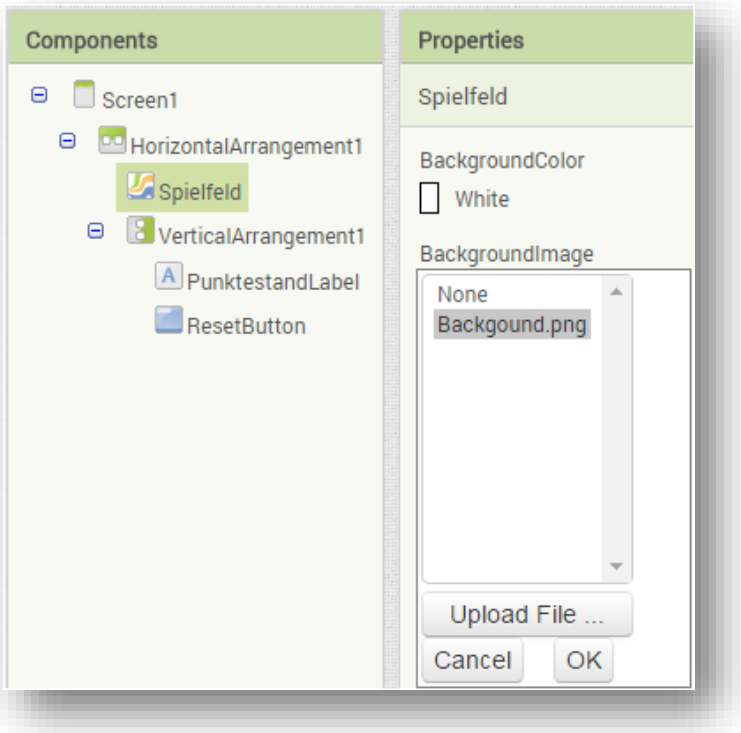
Desktop / InfoSphere goes Android / AngryBlob

Anschließend müsst ihr eurem Canvas den Hintergrund nur noch zuweisen:

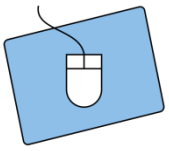


1.) Wählt das Canvas unter Components aus.

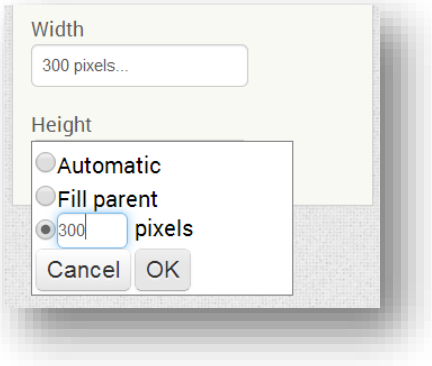
2.) Klickt unter Properties auf *BackgroundImage* und weist dem Canvas das ***background.png*** als Hintergrundbild zu.



Wie wir euch bereits erklärt haben, müsst ihr beim App Inventor manchmal die Größe per Hand einstellen.

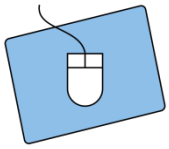


1.) Ihr habt immer noch euer Canvas ausgewählt.
2.) Sucht in den Properties nach ***Width*** (Breite) und ***Height*** (Höhe) und setzt diese jeweils auf 300 pixel.
3.) Testet jetzt eure App und ändert gegebenenfalls die Anzahl der Pixel, damit es auf dem Smartphone perfekt aussieht.



Der Blob

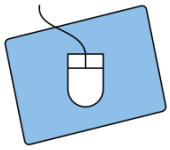
Als nächstes soll der Blob auf das Spielfeld.



1.) Wechselt in der Palette zu *Drawing and Animation.*
2.) Zieht ein ***ImageSprite*** direkt auf die Wiese.
3.) Ändert den Namen des ImageSprites.
4.) Ladet nun aus demselben Ordner den Blob hoch und weist ihn dem ImageSprite zu.

Ein ***ImageSprite*** ist ein besonderes Bild. Dieses kann sich, im Gegensatz zu einem normalen Image, auf einem Canvas bewegen.

Außerdem hat euer Blob eine Position auf dem Spielfeld. Diese könnt ihr in den Properties als ***X- und Y-Koordinaten*** ablesen bzw. verändern.



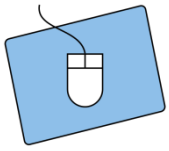
1.) Wählt den Blob unter Components aus.
2.) Ändert unter Properties die X- und Y-Koordinaten und sucht euch eine schöne Startposition für euren Blob aus.

CC BY SA

Der Supercomputer und die Wände

Neben dem Blob braucht ihr noch weitere ImageSprites:

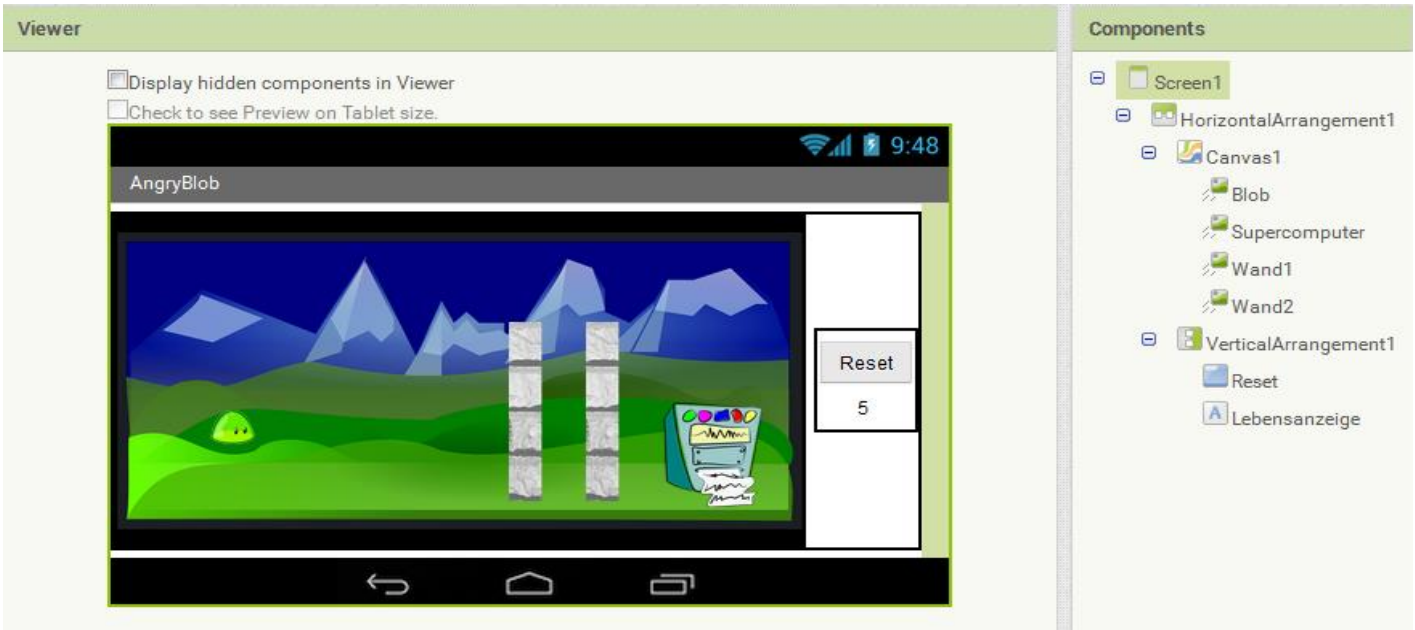
- Ein **ImageSprite** für den Supercomputer und
- **zwei ImageSprites** für die Wände.
    - **Hinweis**: Wenn ihr die Wände drehen wollt, müsst ihr den Wert „**Heading**“ in den **Properties** ändern. Allerdings wird euch das Ergebnis nur auf dem Smartphone angezeigt. (Probiert einmal die Werte 45 und 90 aus.)



1.) Verfahrt genau wie im vorigen Abschnitt und erstellt die ImageSprites an einer sinnvollen Position.
2.) Testet das Gesamtbild eurer App.

Ein kleines Zwischenfazit

So oder so ähnlich sollte eure App jetzt aussehen. Falls ihr noch Fragen habt sprecht einfach kurz mit einem Betreuer.

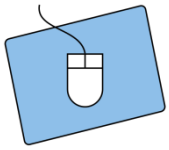


CC BY SA

Angry Blob

Die Bewegung des Blobs

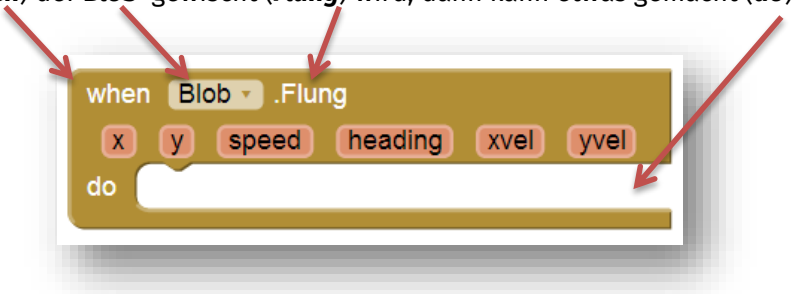
Als erstes soll der Blob durch eure Wischbewegung losfliegen.



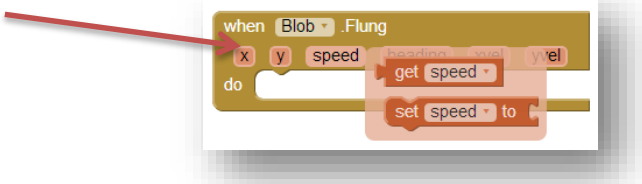
1.) Wechselt in den Blocks-Editor.

2.) Wählt bei eurem ***Blob*** den Block ***when Blob.Flung*** aus.

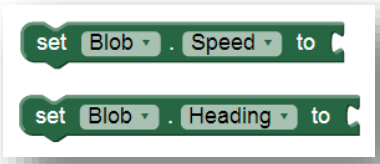
Wenn (***when***) der ***Blob*** gewischt (***Flung***) wird, dann kann etwas gemacht (***do***) werden.



Außerdem werden durch diese Funktion noch Variablen erzeugt die für euch wichtig sind. Nämlich die Geschwindigkeit (***speed***) und die Richtung (***heading***). Diese könnt ihr erhalten, wenn ihr den Mauszeiger auf den Namen der Variablen bewegt:



3.) Jetzt müsst ihr eurem Blob nur noch die neuen Werte für die Geschwindigkeit und die Richtung zuweisen:



4.) Versucht die Blöcke selbständig zu kombinieren und fragt einen Betreuer, falls ihr Hilfe braucht.

5.) Testet die Wischfunktion.

CC BY SA

Auswertung nach jedem Versuch

Der Wurf eures Blobs ist beendet, wenn:

- Er mit einer ***Wand*** kollidiert,
- er mit dem ***Supercomputer*** kollidiert oder
- er den Rand des ***Spielfeldes*** berührt.

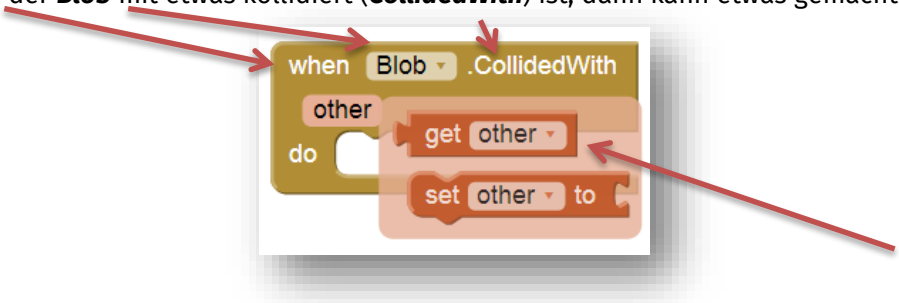
Nach jedem Wurf müssen diese Fälle überprüft werden und anschließend muss entschieden werden, ob weitergespielt wird oder das Spiel zu Ende ist.

Diese Fälle werdet ihr in den folgenden Abschnitten einbauen:

Für die ersten beiden Fälle (**Wand**, **Supercomputer**) könnt ihr die Funktion

when Blob.CollidedWith benutzen:

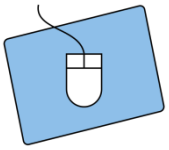
Wenn (***when***) der ***Blob*** mit etwas kollidiert (***CollidedWith***) ist, dann kann etwas gemacht werden.



Zusätzlich erhaltet ihr die Komponente, mit der die Kollision stattgefunden hat. ***Other*** kann also eine ***Wand*** oder der ***Supercomputer*** sein.

Im nächsten Schritt sollt ihr kontrollieren, welcher der drei Fälle eingetroffen ist. Ob der Supercomputer getroffen wurde, könnt ihr folgendermaßen überprüfen:

Falls (***if***) die getroffene Komponente (***other***) der ***Supercomputer*** ist, dann kann etwas gemacht werden.



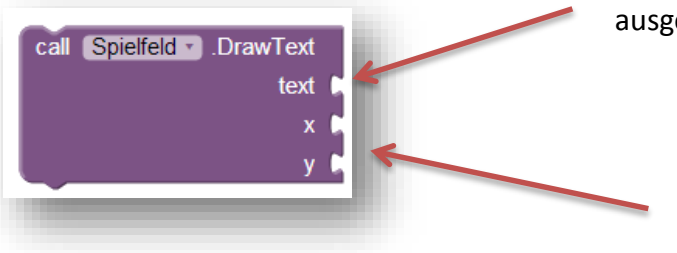
1.) Erstellt das if-then-Konstrukt für die Fälle: Supercomputer und die Wände.

2.) Falls ihr Hilfe braucht, zögert nicht zu Fragen.

CC BY SA

Der Supercomputer wurde getroffen

Falls der Supercomputer getroffen wurde, müsst ihr dem Spieler sagen, dass er gewonnen hat. Dazu könnt ihr einen Text mit folgender Konstruktion auf dem Bildschirm ausgeben:



Hier den Text einhängen, der ausgegebenen werden soll.

Die Position der Textausgabe als Koordinaten auf dem Spielfeld.

Eine Wand wurde getroffen

Falls eine Wand getroffen wurde, müsst ihr folgendes beachten:

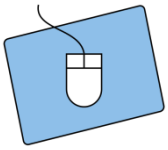
- Ihr müsst ***1 Leben*** abziehen. Dazu braucht ihr folgende Blöcke:

- Ihr müsst die getroffene ***Wand zerstören*** (unsichtbar machen)

- Ihr müsst den Blob an die ***Startposition*** zurücksetzen.

- Und ihr müsst überprüfen, ob der Spieler ***verloren*** hat (Leben = 0)
  - Wenn (***if***) das Leben gleich 0 ist, dann (***then***) gebe einen Text aus, der dem Spieler sagt, dass er verloren hat.

CC BY SA



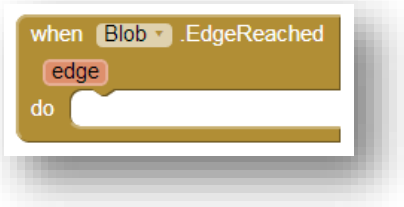
1.) Falls ihr es noch nicht gemacht habt: Implementiert die Treffer für den Supercomputer und die Wände.

2.) Testet eure App.

Super! Bis hier ist es schon ein ganzes Stück Arbeit gewesen. Jetzt kommt der Endspurt ;)

Der Rand wurde getroffen

Jetzt fehlt euch nur noch der Spielfeldrand. Eine Kollision mit diesem könnt ihr über folgende Funktion überprüfen:

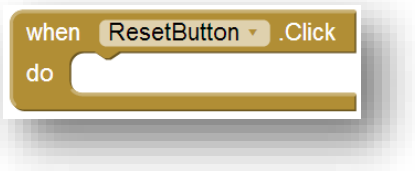


Wenn der Rand getroffen wird, müsst ihr folgendes machen:

- Den Blob auf die **Startposition** zurücksetzen.
- Ein **Leben abziehen**.
- Überprüfen, ob ihr **noch genug Leben** habt.

Der Reset-Knopf

Als letztes müsst ihr dafür sorgen, dass das Spiel ***zurückgesetzt*** werden kann. Dazu braucht ihr den ***Reset-Knopf.*** Wenn der Reset-Knopf gedrückt wird, dann sollen die Grundeinstellungen wiederhergestellt werden.



CC BY SA

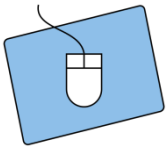
Die Grundeinstellungen

Jetzt müsst ihr euch überlegen, was die Grundeinstellungen sind. Dazu habt ihr folgende Hinweise:

1.) Die Lebensanzeige hat eine Grundeinstellung.

2.) Die Position des Blob muss auf **den Startwert zurückgesetzt** gesetzt werden.

3.) Zerstörte Komponenten (Wände u. Supercomputer) müssen alle **wieder sichtbar** gemacht werden.



1.) Überlegt euch, was alles zurückgesetzt werden muss.

2.) Setzt eure Ideen in der Funktion ***when ResetButton.Click*** um.

Gratulation ☺

Damit habt ihr ein voll funktionsfähiges AngryBlob-Spiel programmiert. Auf der nächsten Seite findet ihr noch Tipps, Hinweise und Anregungen wie ihr das Spiel erweitern könnt. Falls ihr lieber ein neues Spiel programmieren wollt, dann wendet euch an die Betreuer.

CC BY SA

Erweiterungen

Hier findet ihr einige Ideen um das Spiel noch zu erweitern:

Mehrere Wände

Ihr könnt mehrere Wände einbauen, die unterschiedliche Positionen haben. Zusätzlich könnten die Leben erhöht werden, damit es nicht zu schwierig wird.

Bewegte Wände

Ihr könntet die Wände bewegen lassen. Dazu braucht ihr folgendes:

- Ihr müsst euch überlegen, in welchem Bereich sich die Wände bewegen sollen und wann sie ihre Richtung ändern.
  Beispielsweise könntet ihr die Wände immer bis zum Rand laufen lassen. Wenn dann eine Kollision mit dem Rand stattfindet, ändert ihr die Bewegungsrichtung um 180°.

Extra Leben

Ihr könntet einen zweiten Blob auf dem Spielfeld einfügen. Wenn man diesen trifft, bekommt man ein Extra-Leben geschenkt.

Quellenverzeichnis:

 *- Quelle: pixabay.com, Autor: OpenClipartVectors (CC0)*

 *- Quelle: pixabay.com, Autor: geralt (CC0)*

 *- Quelle: pixabay.com, Autor: OpenClipartVectors (CC0)*

 *- Quelle: openclipart.org, Autor: mi_brami (Unlimited Commercial Use)*

 *- Quelle: InfoSphere*

alle weiteren Grafiken sind Screenshots von App Inventor: http://appinventor.mit.edu/explore/

CC BY SA