



UNIVERSIDAD SANTO TOMÁS

PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA

SECCIONAL TUNJA

VIGILADA MINEDUCACIÓN - SNIES 1732



Acreditación Institucional
Internacional

OTORGADA POR EL IAC CINCO ACUERDO 95 DEL 9 DE MAYO-VIGENCIA 5 AÑOS



ACREDITACIÓN
INSTITUCIONAL
DE ALTA CALIDAD
MULTICAMPUS

Vigencia del año 2018



QS STARS
RATED FOR EXCELLENCE



ISO 9001
CERTIFIED



E-Net
CERTIFIED



UNIVERSIDAD SANTO TOMÁS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
SECCIONAL TUNJA

VIGILADA MINEDUCACIÓN - SNIES 1732

División: Ingenierías y Arquitectura
Faculty: Systems engineer
Course: Introduction of Programming
Topic: Fundamentals of software programming

Socializer: Luis Fernando Castellanos Guarín
Email: Luis.castellanosg@usantoto.edu.co
Phone: 3214582098

Topics

- **APPinventor - Game**
- **Course Introduction**
 - What is the software? / Que es el software?
 - Programming languages / Lenguajes de programación
- **History of computing**
- **Concepts and elements of a Flowcharts**



UNIVERSIDAD SANTO TOMÁS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
U N J A
VIGILADA MINEDUCACIÓN - SNIES 1722



Formando personas que transforman





P1T1_way home

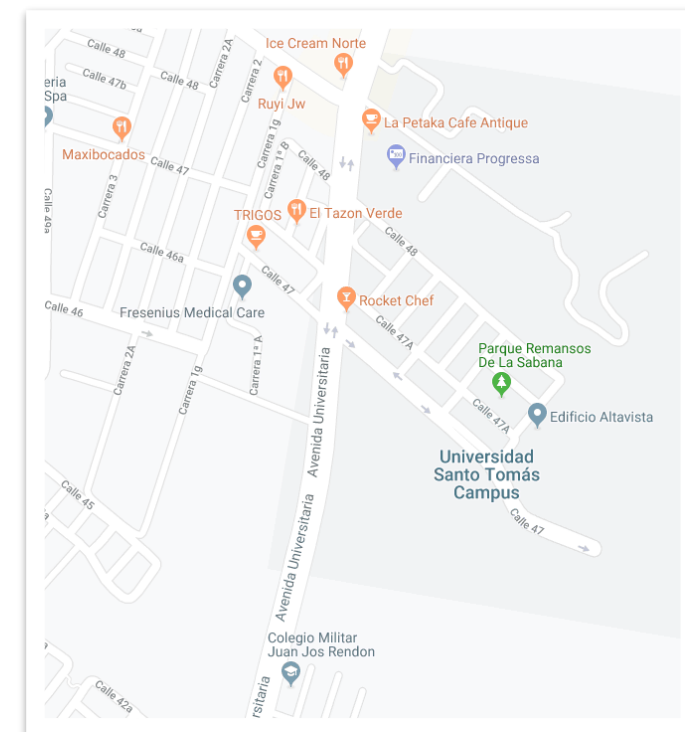
Cada estudiante deberá escribir paso a paso las instrucciones para llegar a su hogar desde la plaza principal de la ciudad/municipio donde vive.

- **NO** se podrá hacer uso de direcciones de la ciudad para ubicar la vivienda
- Deberá describir el recorrido utilizando indicaciones como número de cuadras, lugares representativos, cardinalidad, modos de transporte, entre otros.

P1T1_wayHome_map:

Al finalizar la descripción, los estudiantes intercambiarán su conjunto de instrucciones y con ayuda de Google maps deberán dibujar el recorrido hasta ubicar la vivienda de su compañero.

<https://www.google.com/maps/d/u/0/>



¡Siempre
hacia lo alto!



P1T2_food recipe

En grupos, escribir una receta de comida de forma que una computadora pueda realizar la preparación:

- Ingredientes
- Utensilios
- Pasos para preparar la comida.



¡Siempre
hacia lo alto!



App Inventor

Make a App android

[My Projects ▾](#)[Connect ▾](#)[Build ▾](#)[Settings ▾](#)[Help ▾](#)[My Projects](#)[View Trash](#)[Gallery](#)[Guide](#)[Report an Issue](#)[English ▾](#)[luis.castellanosg@usantoto.edu.co ▾](#)[Start new project](#)[Move To Trash](#)[Publish to Gallery](#)[View Trash](#)

¡Siempre
hacia lo alto!



Palette

Search Components...

User Interface

Button

CheckBox

DatePicker

Image

Label

ListPicker

ListView

Notifier

PasswordTextBox

Slider

Spinner

Switch

TextBox

TimePicker

WebView

Layout

Media

Drawing and Animation

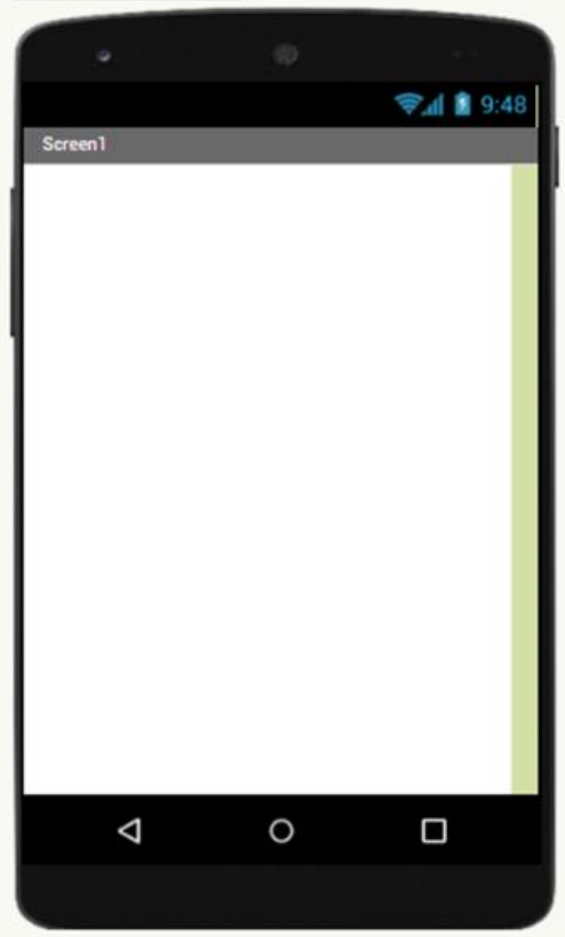
Maps

Sensors

Viewer

☐ Display hidden components in Viewer

Phone size (505,320) ▾



Components

Screen1

Rename Delete

Media

Upload File ...

Properties

Screen1

AboutScreen

AccentColor

AlignHorizontal

AlignVertical

AppName

BackgroundColor

BackgroundImage

BlocksToolkit

CloseScreenAnimation

Icon

OpenScreenAnimation

PrimaryColor

PrimaryColorDark

ScreenOrientation

No es seguro | ai2.appinventor.mit.edu/#6289813069758464

My Projects | Connect | Build | Settings | Help | My Projects | View Trash | Gallery | Guide | Report an Issue | English | luis.castellanosg@usantoto.edu.co |

USTA_2020

Screen1 | Add Screen ... | Remove Screen

Designer | Blocks

Palette

Search Components...

User Interface

Button

CheckBox

DatePicker

Image

Label

ListPicker

ListView

Notifier

PasswordTextBox

Slider

Spinner

Switch

TextBox

TimePicker

WebView

Layout

Media

Drawing and Animation

Maps

Sensors

Viewer

Display hidden components in Viewer

Phone size (505,320)

Screen1

Components

Screen1

Rename | Delete

Media

Upload File ...

Properties

Screen1

AboutScreen

AccentColor

AlignHorizontal

AlignVertical

AppName

BackgroundColor

BackgroundImage

BlocksToolkit

CloseScreenAnimation

Icon

OpenScreenAnimation

PrimaryColor

PrimaryColorDark

ScreenOrientation

componentes
agregados en
la pantalla

componentes
que se pueden
trabajar en la
pantalla

Características
del componente
seleccionado



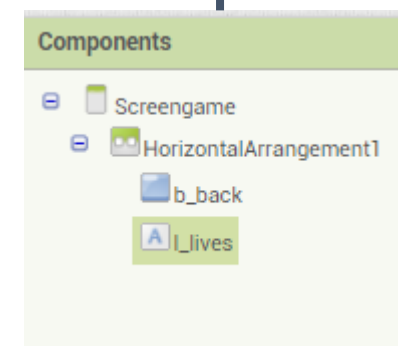
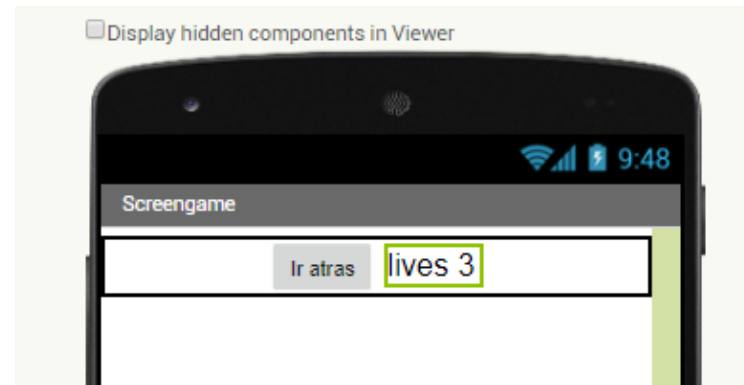
Pasos para crear una APP

1. Definir distribución de las pantallas que se usaran.
2. Diseñar la pantalla iniciar (SCREEN).
3. Crear la distribución de los componentes en la pantalla.
4. Agregar los componentes requeridos.
5. Cambiar nombre de los componentes (definir nombres mejor definidos).
6. Programar las funciones de los componentes.



Create Screen : screengame

1. Insertar un layout: **HorizontalArrangement**, con las siguientes características:
 - a. AlignHorizontal: center
 - b. AlignVertical: center
 - c. BackgroundColor: none
 - d. Height:Automatic
 - e. Width:Fill Parent..
2. Agregar un **button** dentro del layout y cambiar el nombre por **b_back**, en texto "Ir atras"
3. Agregar un label dentro del layout con las siguientes características:
 - a. name: l_lives
 - b. fontsize:20
 - c. text: lives 3

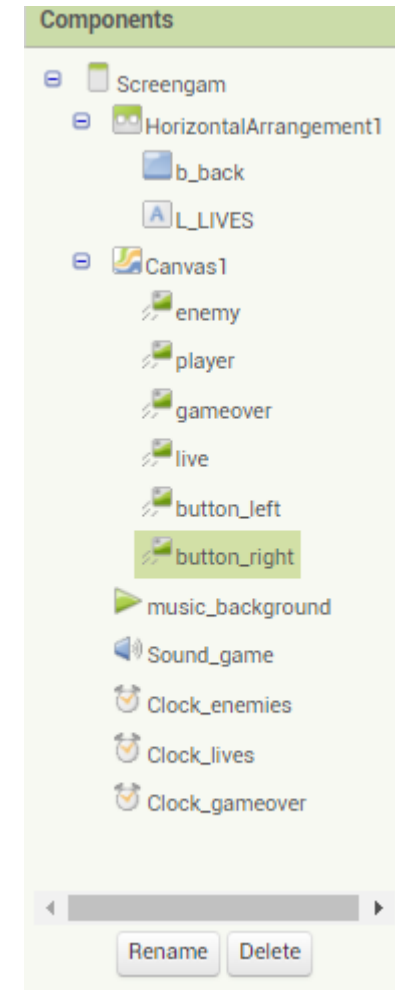
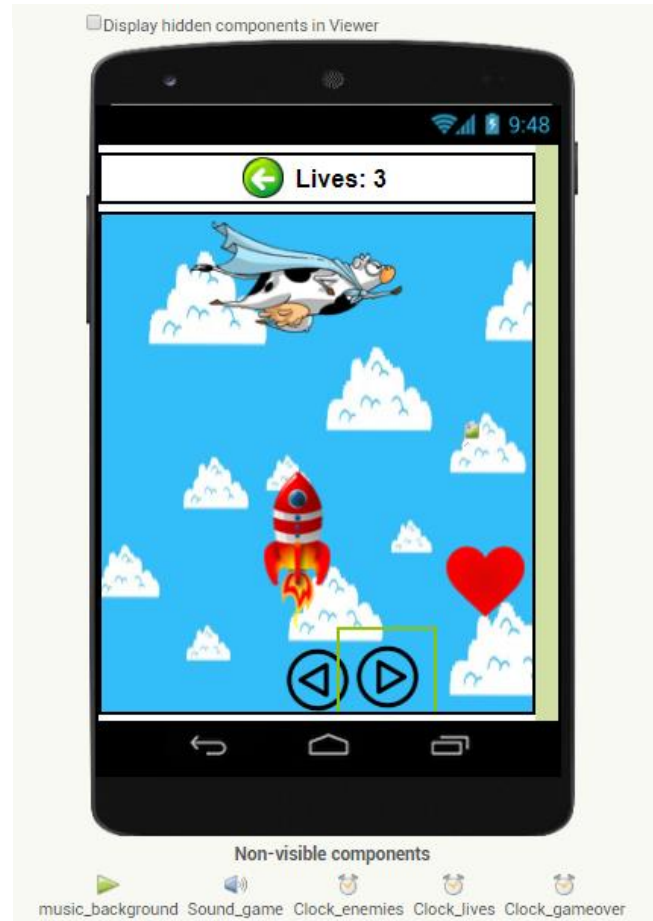


Create Screen : screengame

4. Insertar un canvas que se encuentra en "Drawing and animation":
 - Height & Width = fill parent...
5. Insertar un "imageSprite" y en picture seleccionar la imagen que se tomara para el player
6. Inserte 5 "imageSprite" para:
 - Botton_left
 - Botton_right
 - Enemy.
 - Live
 - GameOver

Cada ImageSprite debe tener cargada una imagen de tipo PNG, menos la de gameover que solo se visualizara cuando las vidas lleguen a cero(0))

7. Agregar tres(3) relojes:
 - Clock_enemies
 - Clock_lives
 - Clock_gameover
8. Agregar elementos para sonidos:
 - Music_background
 - Sound_game



¡Siempre
hacia lo alto!

Create Screen : screengame



initialize global var_lives to 3

to game_over do set Clock_...

to buttons_resize_repositio...

to player_resize_reposition...

to live_resize_reposition d...

to enemy_resize_reposition ...

when enemy .CollidedWith ...

when Clock_gameover .Timer...

when Clock_enemies .Timer ...

when Clock_lives .Timer do...

when enemy .EdgeReached ...

when live .EdgeReached e...

when button_left .Touched ...

when button_right .Touched...

when live .NoLongerCollidi...

when b_back .Click do open...

when Screengam .Initialize...



**Estos son los bloques que se requieren para crear el nivel 1
Del videogame.**

A continuación se explica cada uno de ellos.



**¡Siempre
hacia lo alto!**

Create Screen : screengame / blocks – procedures resize and reposition

Como las pantallas de los celulares tienen diferentes dimensiones, debemos redimensionar todos los elementos del juego a cada pantalla

```
to buttons_resize_reposition
do
  set button_right . Width to Canvas1 . Width × 0.15
  set button_right . Height to Canvas1 . Height × 0.15
  set button_right . X to Canvas1 . Width × 0.5
  set button_right . Y to Canvas1 . Height - button_right . Height × 2
  set button_left . Width to Canvas1 . Width × 0.15
  set button_left . Height to Canvas1 . Height × 0.15
  set button_left . X to Canvas1 . Width × 0.5 - button_left . Width
  set button_left . Y to Canvas1 . Height - button_left . Height × 2
```

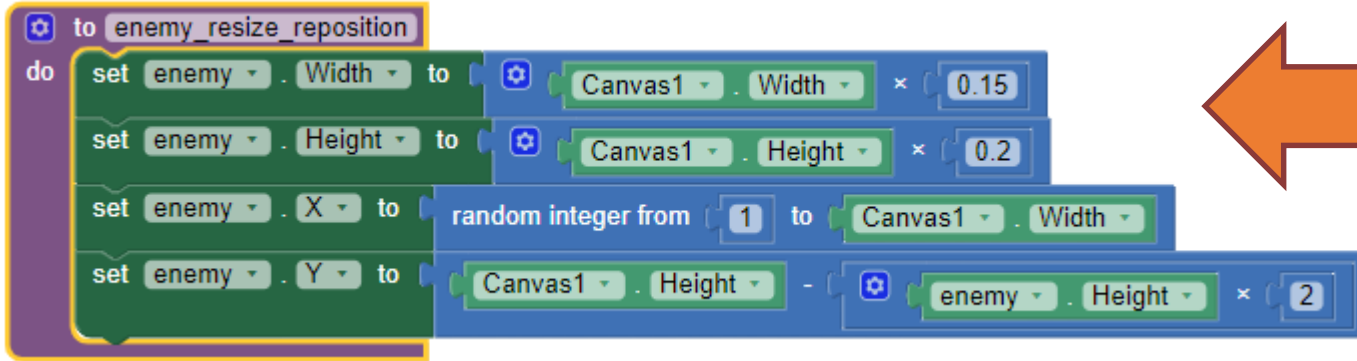
Se redimensionan los botones y se ubica en el centro de la pantalla en la parte inferior.

```
to player_resize_reposition
do
  set player . Width to Canvas1 . Width × 0.25
  set player . Height to Canvas1 . Height × 0.2
  set player . X to Canvas1 . Width × 0.5 - player . Width × 0.5
  set player . Y to 1
```

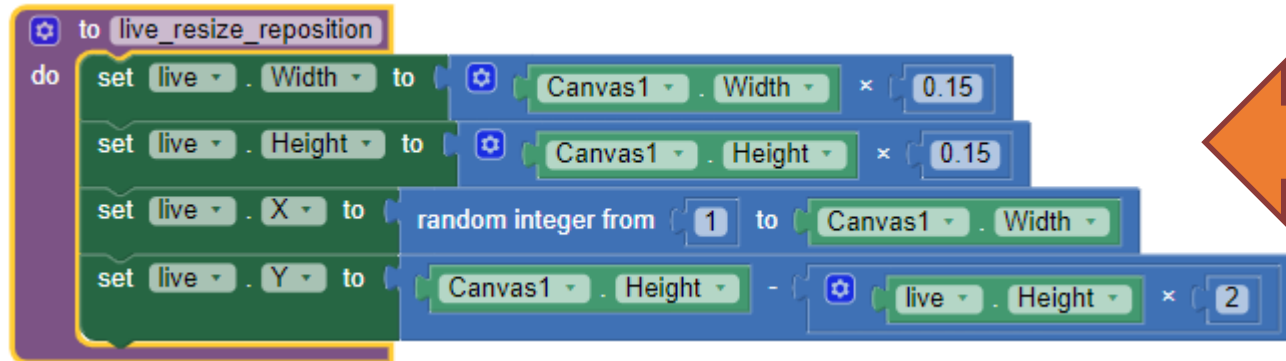
Se redimensiona el player y se reubica en el centro en la parte superior de la pantalla

¡Siempre
hacia lo alto!

Create Screen : screengame / blocks – procedures resize and reposition

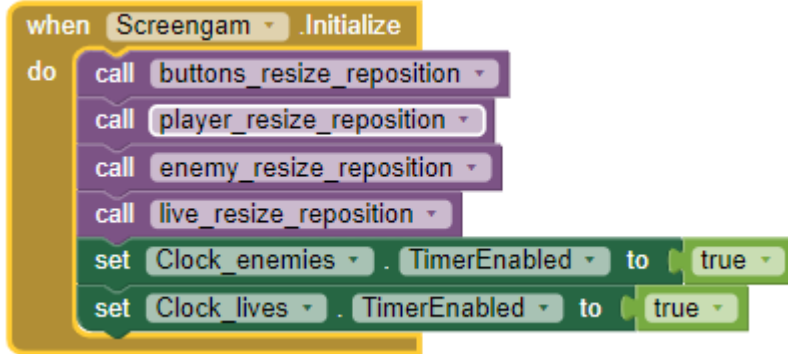


Se redimensionan el enemigo y se ubica en el centro de la pantalla en la parte inferior (donde cada vez que se cree tomara una posición aleatoria en X)



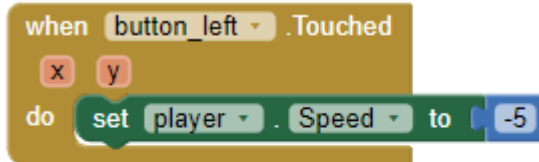
Se redimensionan la vida y se ubica en el centro de la pantalla en la parte inferior (donde cada vez que se cree tomara una posición aleatoria en X)

Create Screen : screengame / blocks – Player move left and right



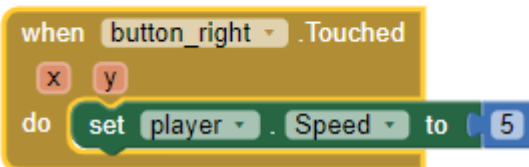
```
when Screengame .Initialize
do
  call buttons_resize_reposition
  call player_resize_reposition
  call enemy_resize_reposition
  call live_resize_reposition
  set Clock_enemies . TimerEnabled to true
  set Clock_lives . TimerEnabled to true
```

Scratch code block for screen initialization. It contains a 'when Screengame .Initialize' event block followed by a 'do' loop containing: 'call buttons_resize_reposition', 'call player_resize_reposition', 'call enemy_resize_reposition', 'call live_resize_reposition', 'set Clock_enemies . TimerEnabled to true', and 'set Clock_lives . TimerEnabled to true'.



```
when button_left .Touched
do
  set player . Speed to -5
```

Scratch code block for left button touch. It contains a 'when button_left .Touched' event block followed by a 'do' loop containing: 'set player . Speed to -5'.



```
when button_right .Touched
do
  set player . Speed to 5
```

Scratch code block for right button touch. It contains a 'when button_right .Touched' event block followed by a 'do' loop containing: 'set player . Speed to 5'.

Cuando se inicialice la pantalla se llamaran los procedimientos que permites redimensionar y ubicar los elementos del videojuego. A su vez se habilitan los relojes de enemigos y de vidas, con ellos se controla la velocidad de movimiento.

- Cuando el usuario toque el botón izquierdo se le cambiara a negativo la velocidad en X en este caso se determina en -5
- Cuando toque el botón derecho se le asignara positiva la velocidad en X en este caso se determina en 5

Create Screen : screengame / blocks – Clock enemies and lives

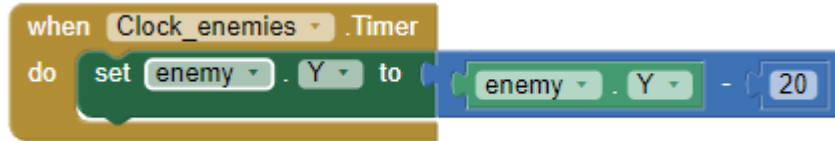
Properties

Clock_enemies

TimerAlwaysFires
☒

TimerEnabled
☐

TimerInterval
500



El reloj para enemigos se configuro con una frecuencia de 500 milisegundo donde en cada instancia se moverá en 20 pixeles hacia arriba

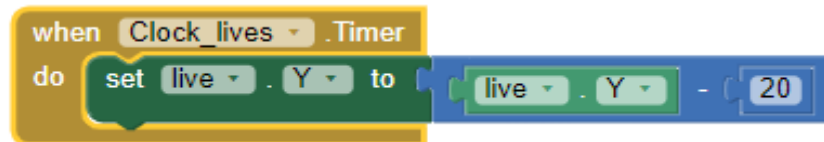
Properties

Clock_lives

TimerAlwaysFires
☒

TimerEnabled
☐

TimerInterval
2000



El reloj para vidas se configuro con una frecuencia de 2000 milisegundo donde en cada instancia se moverá en 20 pixeles hacia arriba (cuatro veces más lento que los enemigos)

¡Siempre
hacia lo alto!

Create Screen : screengame / blocks – control of lives

initialize global var_lives to 3

Para el control de las vidas se crea una variable global que se denominara “var_lives”

```
when enemy . CollidedWith
  other
  do
    if
      player = get other
    then
      call player_resize_reposition
      call enemy_resize_reposition
      set global var_lives to get global var_lives - 1
      set L_LIVES . Text to join " Lives "
      get global var_lives
    if
      get global var_lives ≤ 0
    then
      call game_over
```

Cuando el “**enemy**” colisiona con el “**player**” se descuenta una vida de la variable “var_lives” y para que el usuario visualice este cambio se modifica el texto en el label “L_LIVES”

En caso que el número de vidas se igual o menor que cero se ejecutara el procedimiento “game_over”

```
to game_over
do
  set Clock_enemies . TimerEnabled to false
  set Clock_lives . TimerEnabled to false
  set gameover . Picture to "gameover.png"
  set gameover . Height to Canvas1 . Height × 0.5
  set gameover . Width to Canvas1 . Width
  set Clock_gameover . TimerEnabled to true
```

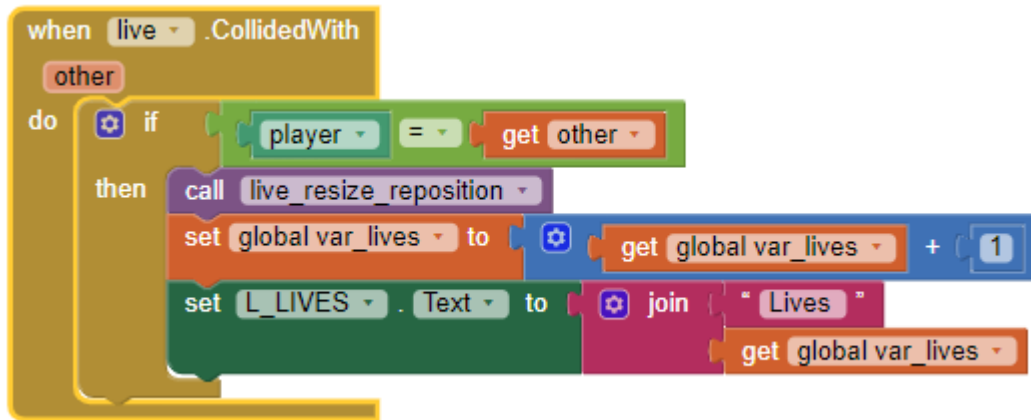
Inhabilitar los relojes, cargar la imagen de “GAME OVER” hacer que la imagen se expanda en toda la pantalla. Habilitar el reloj de game over.

```
when Clock_gameover . Timer
do
  close screen
```

El reloj “clock_gameover” esta configurado a 2000 milisegundo: una vez pase ese tiempo se cerrara la ventana.

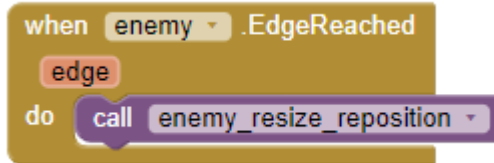


Create Screen : screengame / blocks – control of lives



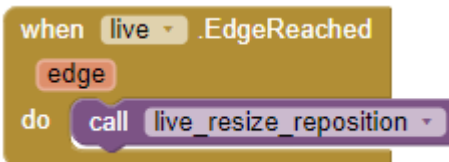
```
when live .CollidedWith  
  other  
  do  
    if player = get other  
    then  
      call live_resize_reposition  
      set global var_lives to get global var_lives + 1  
      set L_LIVES .Text to join "Lives " get global var_lives
```

Cuando el “**live**” colisiona con el “**player**” se suma una vida a la variable “var_lives” y para que el usuario visualice este cambio se modifica el texto en el label “L_LIVES”



```
when enemy .EdgeReached  
  edge  
  do  
    call enemy_resize_reposition
```

Cuando el “**enemy**” llegue a la parte superior de la pantalla sin colisionar con el “**player**”, se reubica en la parte inferior de la pantalla nuevamente para generar un nuevo reto al usuario.



```
when live .EdgeReached  
  edge  
  do  
    call live_resize_reposition
```

Cuando el “**live**” llegue a la parte superior de la pantalla sin colisionar con el “**player**”, se reubica nuevamente en la parte inferior de la pantalla.

¡Siempre
hacia lo alto!