



# Tecnológico de Monterrey

Proyecto de Desarrollo de Videojuegos

Doctor Iván Olmos Pineda

Enero 2016 Grupo 1

Armando Canto García A01322361

Luis Alfredo León Villapún A01322275

Eduardo Linares Figueroa A01322803

## **Pato ALV**

La traducción de las siglas ALV es “A Libre Vuelo”, ya que este juego fue creado por la iniciativa de las mascotas del ITESM PUE, las cuales son los patos; dentro de nuestro juego el usuario será un pato, el cual es el personaje principal de todo el juego, el usuario debe enfrentarse a diferentes niveles en los cuales se encontrara con obstáculos inmóviles como también enemigos los cuales cuentan con inteligencia artificial, estos serán diferentes personajes relacionados con el ITESM, como los estudiantes y jugadores de equipos representativos. Dentro de los retos, el jugador debe recoger elementos dentro de la interfaz, estos pueden ser monedas, las cuales suman puntuación al jugador, u objetos que son habilidades especiales que ayudan al jugador a superar los niveles con mayor facilidad.

La interfaz seleccionada para la realización de este proyecto es Game Maker, la cual usa un lenguaje llamado GML (Game Maker Language), pero también es posible usar la interfaz gráfica, la cual como App Inventor se basa en acciones y en objetos, los cuales como la teoría en clases, son los agentes dentro de nuestro videojuego.

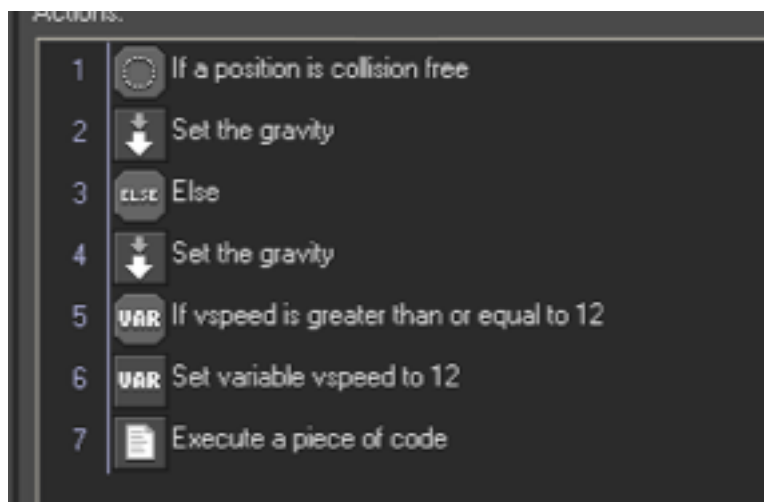


Dentro de la clase, solo vi el motor gráfico de GTGE, por lo cual para la implementación de nuestro proyecto, aprendimos por medio de videotutoriales las funciones básicas como también avanzadas de este motor, y a la vez el análisis de diferentes ejemplos y juegos como tal para poder implementar ciertas funciones dentro de nuestro juego.

### Código:

Primero dentro de la interfaz se creó el objeto el pato, en el cual se dividen los sprites dependiendo de donde se encuentra viendo o hacia qué dirección el usuario mueve al pato, se inicializa viendo hacia la derecha, este pato tiene la habilidad de saltar, para lo cual es necesario implementar la gravedad, GameMaker(GM) tiene una función de gravedad, por lo cual solo dentro de las restricciones hacia el objeto pato, se inicializa está cada vez que el pato deja el piso, y así provoca que regrese al suelo, y una vez que el pato ha tocado el suelo, este debe reiniciar su gravedad a suelo, si no, el pato seguiría cayendo dentro del ambiente en el cual es inicializado.

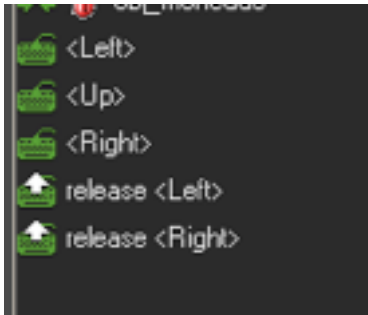
En esta parte de código, por medio de la interfaz grafica se define la gravedad:



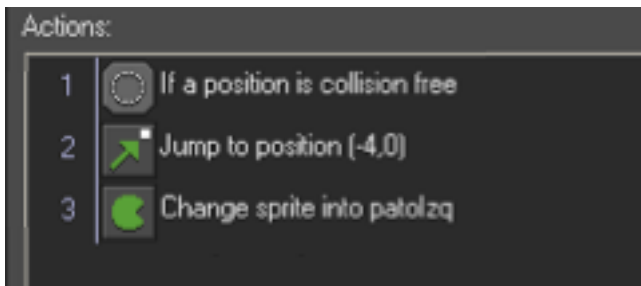
Primero definimos que si nuestro objeto, es este caso el pato, no se encuentra con ninguna colisión, ya se el piso o algun otro objeto, es decir se encuentra libre en el ambiente, la gravedad obtenga un valor y cuando llegue a una colision el valor se la gravedad se vuelve 0,

para que nuestro objeto deje de caer en el ambiente.

Dentro de GM, se definen las acciones de cada tecla que sea seleccionada para el juego, tenga su funcion predefinida .

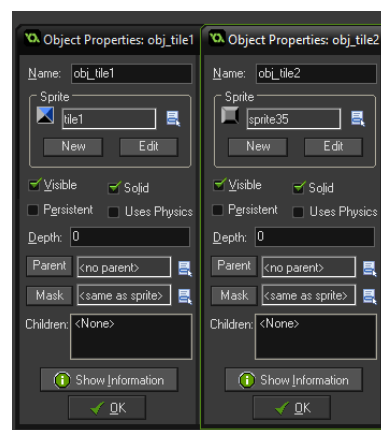
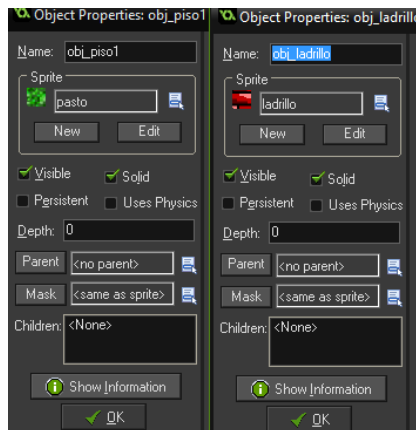


Para nuestro agente pato, se tiene las teclas de arriba, derecha e izquierda, las cuales todas al momento de ser accionadas cambian el sprite de nuestro agente.

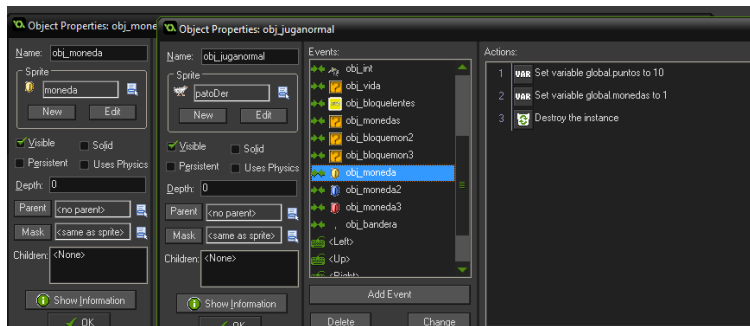


Aquí es el caso de la tecla izquierda, en la cual, por ser movimiento en x, revisa que cuando no hay colision mueva el agente hacia la izquierda 4 pixeles.

Luego creamos los objetos inanimados, los cuales hacen referencia al ambiente del juego, para nuestro juego creamos dos objetos los cuales cuenta con la colisión hacia el pato como también hacia los enemigos, los cuales son el piso, para primero el nivel es pasto y en el segundo son bloques metálicos, los dos de 32x32 pixeles, y con los bloques es lo mismo, en el segundo nivel se usó un bloque metálico.

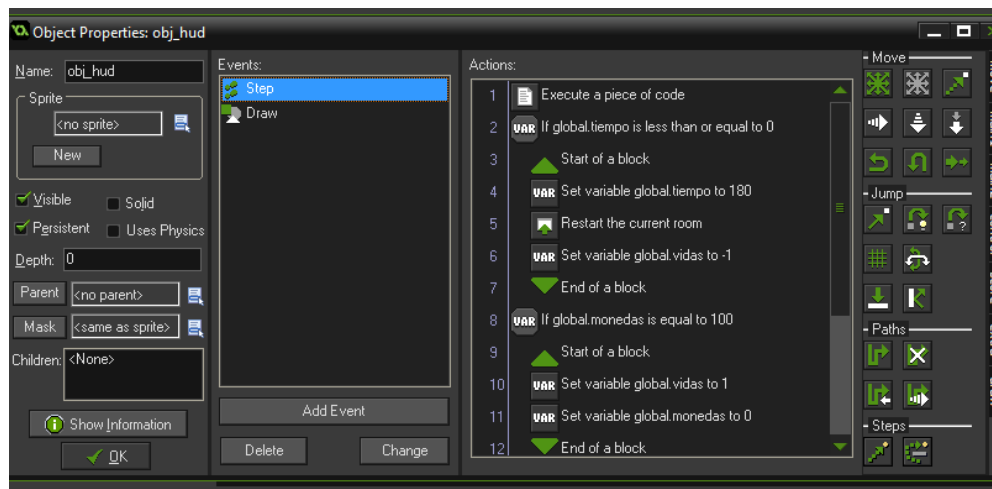


También dentro del juego tenemos “ítems” los cuales ayudan al jugador aumentando su puntuación o en el caso del ítem de la monster borrego le da una vida más al pato, estos objetos también tienen colisión al pato, y al crearse la colisión el objeto desaparece.

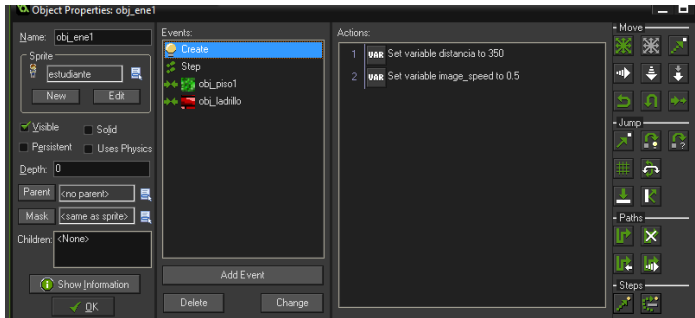


aquí se ve cómo es que el objeto moneda interactúa con el pato.

Por lo tanto, creamos también el objeto HUD, el cual lleva control de los puntos como vida de nuestro jugador, este se inicializa con variables globales las cuales dependiendo del juego y por medio de la interacción del jugador con el ambiente, realiza los cambios y los imprime en pantalla.



Para la implementación de los enemigos, creamos dos tipos, el primero es un agente que se puede decir tonto, ya que no cuenta con ninguna implementación de inteligencia artificial, por lo tanto, solo tienen instrucciones de moverse en x, y con cierta velocidad y límites preestablecidos.



Primero se crea el objeto con el cual se ponen sus restricciones en este caso, que tenga colisión con los bloques y el piso.

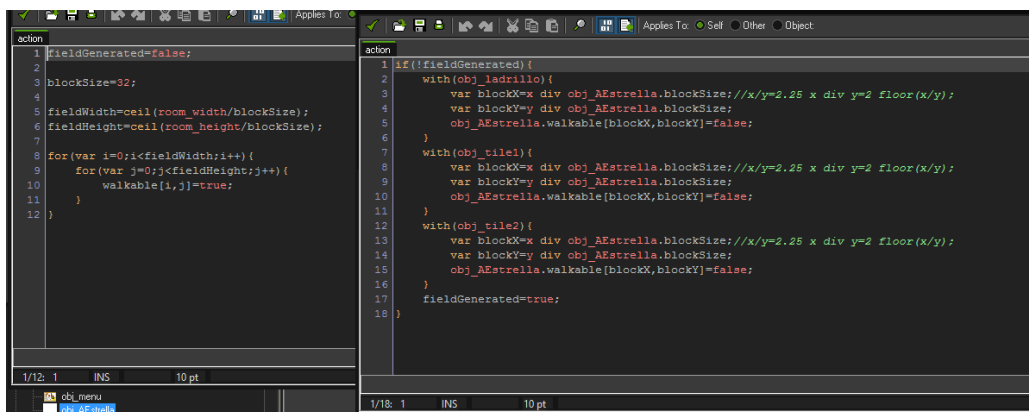
Como también a la hora de que es generado, este tenga una

velocidad en x la cual lo permita moverse de manera “tonta” dentro del ambiente.

## Implementación A\*

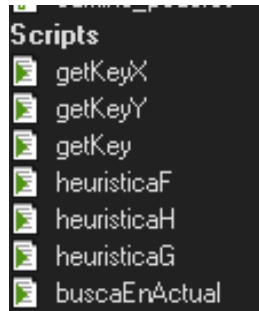
GameMaker, tiene una función propia la cual genera el algoritmo de A\* para los objetos, pero nosotros por medio de investigación encontramos una implementación iterativa del algoritmo.

Tenemos primero el objeto AEstrella el cual crea un mapeo del room, dividiendo todo el room en bloques de 32x32 pixeles, ya que nuestro room de cada nivel está dividido de la misma forma.

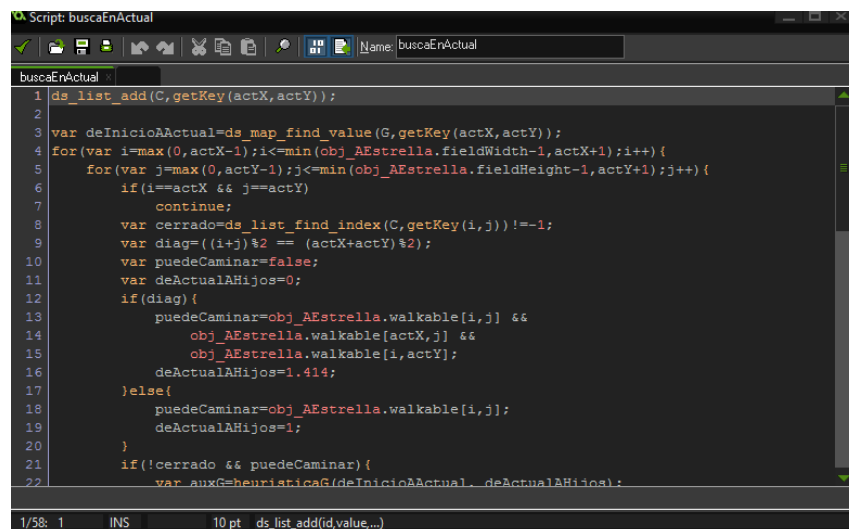


Una vez que se tiene el mapeo del room por medio defunciones tenemos x y y, y dentro de GameMaker creamos los scripts los cuales son:

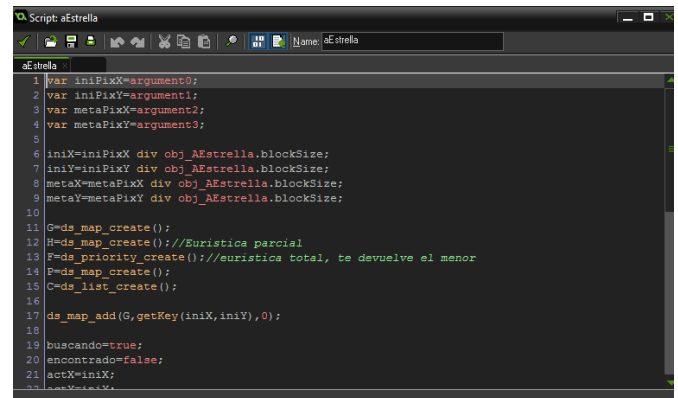
- Heurística G
- Heurística H
- Heurística F



Como su nombre lo menciona calculan estas distancias para ser usadas en el script de búsqueda actual, el cual es nuestro BFS, y este BFS fue definido con profundidad uno, porque al poner mayor nivel de profundidad se vuelve un juego más complejo hacia el usuario y por lo tanto pierde su temática de arcade.



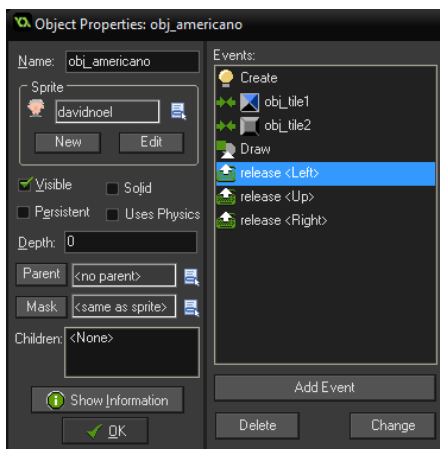
Después de tener nuestro BFS, este es llamado en nuestro script de a estrella el cual lo analiza en todos los nodos y se hace el algoritmo de A\* como tal.



```
1 var iniPixX=argument0;
2 var iniPixY=argument1;
3 var metaPixX=argument2;
4 var metaPixY=argument3;
5
6 iniX=iniPixX div obj_AEstrella.blockSize;
7 iniY=iniPixY div obj_AEstrella.blockSize;
8 metaX=metaPixX div obj_AEstrella.blockSize;
9 metaY=metaPixY div obj_AEstrella.blockSize;
10
11 G=ds_map_create();
12 H=ds_map_create();//Euristicas parcial
13 F=ds_priority_create();//euristica total, te devuelve el menor
14 F=ds_map_create();
15 C=ds_list_create();
16
17 ds_map_add(G,getKey(iniX,iniY),0);
18
19 buscando=true;
20 encontrado=false;
21 setX=iniX;
22 setY=iniY;
```

## Agente Inteligente

Para nuestro Agente inteligente, seleccionamos a David Noel, un sprite un poco más grande que nuestro jugador el cual por medio de A\* persigue al jugador dentro del ambiente del Juego.



Este objeto como los agentes tontos, cuenta con sus respectivas colisiones hacia el ambiente, pero a la vez tiene la implementación de seguir a nuestro jugador dependiendo que dirección tenga. Es decir, el camino trazado por nuestra A\* se crea gracias al movimiento del jugador dentro del

ambiente.

Se define el camino llamando el objeto A\* para el agente inteligente.