

Algoritmos y Estructuras de Datos: Especificación del Trabajo práctico

2do Cuatrimestre 2010

Mortal Igo 2



Docente: Alejandro Frankel

Ayudantes: Ariel Alvarez, Daniela Soto Perez

Mortal Igo 2

Introducción:

En este tp se les pedirá realizar un bot para el juego “Mortal Igo 2”. Es decir, deberán desarrollar una aplicación capaz de jugar de una manera más o menos inteligente y, eventualmente, ganar. Para esto contarán con el juego propiamente dicho, el cual será una caja negra (no necesitan saber cómo funciona internamente salvo por las especificaciones que se les dará), así como con una unit de pascal que les proveerá de funciones para enviarle órdenes al jugador que controlará el bot.

El Juego:

Instalación:

En una primer instancia, la instalación del juego deberá realizarse mediante los pasos siguientes. Ante cualquier inconveniente se les proveerá de una máquina virtual con las herramientas necesarias funcionando, listas para desarrollar el tp.

En windows:

Para correr el juego se necesitan 4 cosas:

1. Python2.6: <http://www.python.org/download/releases/2.6/>
2. Pygame (biblioteca para desarrollo de juegos):
<http://pygame.org/ftp/pygame-1.9.1.win32-py2.6.msi>
3. Numpy (biblioteca para cálculos científicos):
<http://sourceforge.net/projects/numpy/files/>
4. El código fuente del juego, que pueden encontrar acá:
<http://code.google.com/p/mortal-igo/downloads/detail?name=mortal-igo-1.0.1.rar&can=2&q=#makechanges>

En linux:

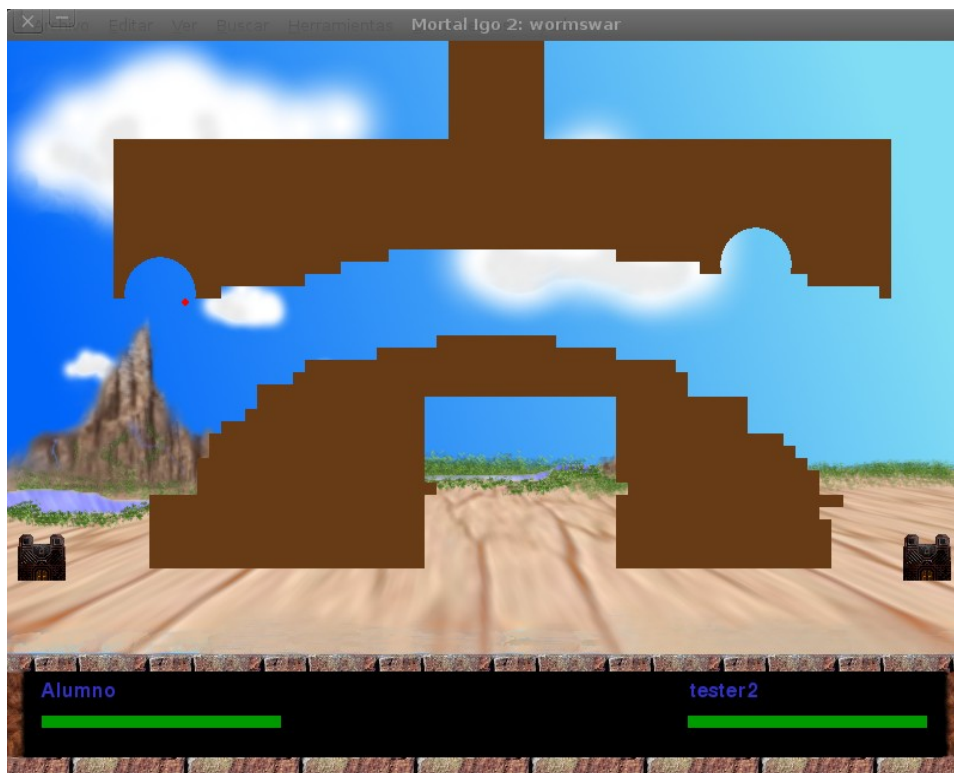
Los 1,2 y 3 de windows se encuentran en los repositorios de la mayoría de las distribuciones linux. El cuarto elemento es independiente de la plataforma.

Habiendo instalado 1,2 y 3 en ese orden, ejecutan el juego haciendo doble click en “start.py”. Aparecerán dos ventanas, una consola por donde se ingresarán las órdenes para el jugador humano, y otra en donde se verán las acciones de los jugadores. Si todo funcionó correctamente, esta segunda ventana mostrará el mensaje “esperando a que el bot se conecte”.

Cómo es:

El juego posee una mecánica muy sencilla. Hay dos castillos, uno a cada extremo de la pantalla, que se irán disparando proyectiles (léase, puntitos rojos) hasta que la vida de alguno de los dos se acabe. Aquel que permanezca en pie será el ganador.

Entre los jugadores habrá ciertos obstáculos, que se deformarán a medida que los proyectiles impacten contra ellos, con lo cual parte de la estrategia del juego se basa en horadar un camino lo más rápidamente posible entre los obstáculos para poder alcanzar al contrincante.



El jugador de la derecha será siempre el bot desarrollado por ustedes.

Los obstáculos están distribuidos según un mapa 'ejemplo.map' presente en el directorio 'maps' que encontrarán junto con el código del juego. Este es un archivo de texto con caracteres '#' dispuestos de manera tal que cada uno de ellos corresponde a un bloque compuesto por varios puntos (sus dimensiones se especifican en el anexo).

La unit botfunctions:

Su nombre es 'botfunctions' y les brindará dos funciones y dos tipos de datos:

```

type
  TPunto=record
    x:integer;
    y:integer;
  end;

  TDisparoResult=record
    estado:string[16]; {OK | WIN | FAIL}
    enemyColision:TPunto; {punto en el cual colisiona el proyectil enemigo}
  end;

function IniciarBot(nombre: string):TBOT;
function Disparar(var miBot:TBOT; angulo:integer; potencia:integer):TDisparoResult;
```

La función **IniciarBot** recibe el nombre que tendrá el jugador controlado por el bot, y retorna un bot handler. Esta función se invoca **una única vez**, ya que es la que inicia la comunicación con el juego. En caso de que el juego esté corriendo en una pc remota, puede llamarse a esta función con un parámetro extra, que será de tipo string y tendrá la IP del host remoto (por ejemplo '190.193.150.235').

La función **Disparar** será la que le indique al jugador controlado por el bot el ángulo y potencia de disparo. Recibirá un bot handler (que será único durante toda la aplicación en pascal), el ángulo y la potencia del proyectil. Esta función es bloqueante, lo cual significa que se quedará esperando hasta recibir una respuesta del juego el tiempo que sea necesario. En caso de haber derrotado al oponente, devolverá un registro de tipo TDisparoResult con su campo 'estado' con el string 'WIN'. En caso de haber sido derrotado, retornará el string 'FAIL'. En cualquier otro caso, devuelve estado con 'OK' y enemyColision con la posición (x;y) en la cual el proyectil enemigo colisionó (esto permite mantener actualizado el mapa, como se verá más adelante).

Esta unidad la encontrarán en el directorio 'uses'. Allí mismo encontrarán un archivo 'ejemplo.pas' que utiliza botfunctions (nota: para que funcione correctamente, el juego ya debe encontrarse en ejecución antes de correr el ejemplo).

El Bot:

El bot será una aplicación desarrollada íntegramente en pascal y compilada con free-pascal, que quedará a cargo del alumno y constituye el TP propiamente dicho. Se servirá de las funciones provistas por botfunctions para interactuar con el juego, y levantará el mapa correspondiente accediendo al archivo de texto 'ejemplo.map' dentro del directorio 'maps'.

Esta aplicación deberá encontrarse dentro del directorio 'uses', y accederá al mapa sin cambiarlo de lugar.

El mapa 'ejemplo.map' puede ser libremente modificado para asegurarse de contemplar toda posible situación.

Es un requerimiento de aprobación recalcular la potencia y ángulo de disparo en cada turno, ya que transcurrido el mismo es posible que el mapa se haya modificado y así mismo la trayectoria óptima.

Se recomienda modularizar la resolución mediante units.

Primera Entrega: (fecha a acordar)

Deberán aplicar conceptos de vectores y matrices y hacer uso de al menos un algoritmo visto en clase (ordenamiento, búsqueda de máximos/mínimos, etc.) e indicarlo.

La entrega será vía mail, enviando diagrama y codificación de la solución (sin ejecutables, sólo el código). El diagrama deberá estar en formato pdf o png.

El asunto del mail tiene que respetar el formato:

TP ENTREGA 1 <curso> <número de grupo>

Y será enviado a alvarez_ariel_utn@yahoo.com.ar con copia a kaos_mundial@hotmail.com. Antes de las 48 horas tienen que recibir una respuesta. De no suceder, reenvíen el tp.

Segunda Entrega: (fecha a acordar)

Idem primer entrega pero utilizando estructuras dinámicas para el almacenamiento de la información del mapa.

Anexo:

Datos necesarios:

- **Los ejes x e y tienen su origen en la esquina superior izquierda y crecen hacia la esquina inferior derecha.**
- Tamaño de la pantalla de juego: ancho=800, alto=500
- Gravedad: 0,7 unidades por tick de clock al cuadrado.
- Dimensiones de los jugadores: 40x40
- Ubicación del bot (punto correspondiente a su esquina superior): (10;400)
- Ubicación del jugador 2 (punto correspondiente a su esquina superior): (750;400)
- Cada carácter # en el mapa corresponde a un cuadrado de lado 10.
- El **mapa de obstáculos** sólo se extiende en un rectángulo que **comienza en (60;0) y termina en (740;400)**
- Cada colisión contra un punto obstáculo provoca una explosión que destruye todo lo que se encuentre a una **distancia menor o igual de 30 unidades** del punto de colisión.
- El proyectil se considera puntual (no tiene cuerpo).
- Cada jugador tiene 100 de vida y cada golpe le resta 20.
- Ante la necesidad de convertir un valor real a entero, se trunca, no se redondea.

Software:

Free-Pascal Compiler: <http://www.freepascal.org/download.var>

Todo el código provisto en el tp está pensado para funcionar tanto en windows como en cualquier distribución de linux, siempre y cuando se cumplan las dependencias indicadas.

**Cualquier duda sobre enunciado pregunten por el grupo, o bien a alvarez_ariel_utm@yahoo.com.ar → Ayudante Ariel Alvarez
dfm.sotoperez@yahoo.com.ar → Ayudante Daniela Soto perez**

FAQ:

En windows me aparece “import main not found” o algo similar:

Probablemente windows no haya puesto PYTHONPATH en el PATH del sistema. Esto puede hacerse manualmente siguiendo estas instrucciones:

To augment PYTHONPATH, run regedit and navigate to HKEY_LOCAL_MACHINE\SOFTWARE\Python\PythonCore and then select the folder for the python version you wish to use. Inside this is a folder labelled PythonPath, with one entry that specifies the paths where the default install stores modules. Right-click on PythonPath and choose to create a new key. You may want to name the key after the project whose module locations it will specify; this way, you can easily compartmentalize and track your path modifications.

Your new key will have one string value entry, named (Default). Right-click on it and modify its value data; this should be text in the same format as the Path environment variable discussed above--absolute directory paths, separated by semicolons. If one project will use modules from several directories, add them all to this list. (Don't bother attempting to add more string value entries to your new key, or to the original PythonPath key, since they will be ignored.) Once these new registry entries are in place, your scripts' import statements should work fine.