Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

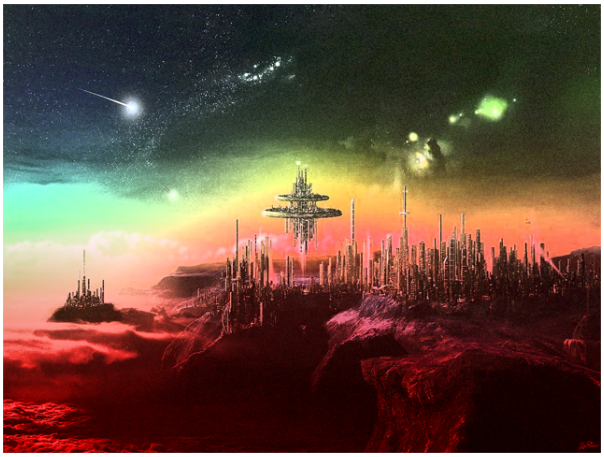
IIC2233 Programación Avanzada (II/2014)

Tarea 4

1. Objetivos

- Aplicar conceptos y nociones de delegates, eventos y desarrollo de una interfaz gráfica en WPF.
- Modelar un problema en base a la idea de Back-End/Front-End.

2. Introducción



No es un buen momento para estar viviendo en Adrianus LXIX. Tampoco lo han sido los últimos años y menos aun los que están por venir. Es increíble que finalmente haya logrado descifrar las escrituras de Lord Enzio; con la actual situacion del gobierno del General Zod-To cualquiera se hubiese rendido. Abandoné mi vida para poder llevar a cabo este maldito trabajo... maldito – sí – pero también totalmente necesario. Durante estos largos años, todos los días miré desde las alturas mi ciudad. Tan majestuosa antes de la caída de Fadic y ahora tan solo una prisión que apesta a muerte y desesperación. Esa imagen es lo único que me da la energía para seguir adelante con esto... Tengo miedo, mucho miedo – no voy a mentir- pero estoy tranquilo a la vez... Hoy se marca el comienzo de algo sin precedentes. Apenas haga conocidas las confesiones de Lord Enzio, Adrianus entrará en guerra. Nunca antes ha ocurrido, lo sé... pero esta vez es distinto, en estos últimos años nuestra especie ha cambiado demasiado. Tanta represión ha cultivado algo más grande dentro de nosotros, tan grande que sobrepasa las dificultades que tendría una especie como la nuestra para pelear.
Esta carta la escribo con el propósito de... de tan solo escribir... He vivido muchos años solo desde que encontré la carta de Lord Enzio y hoy sentía la necesidad de comunicar mis pensamientos. Querido lector, espero que algún día encuentres esta carta, y espero que para cuando la leas, el mundo esté nuevamente en equilibrio. Quiero que sepas que todo comenzó aquí... todo. Sin importar lo que pase de ahora en adelante, para bien o para mal, todo recaerá en este punto en la historia... La filtración de las confesiones Enzianas!

Conde Halepi Ment-Hel

3. Problema

Se deberá realizar un programa en WPF que implemente la verdadera guerra en Adrianus LXIX. Debido a que los Adrianinos son una especie de gusano, deberas implementar un juego basado en la idea de worms[1]. Tu programa consistirá en un juego por turnos para 2 jugadores. Cada jugador posee un ejército de worms con ciertos atributos y un arsenal con el cual pueden atacar. En cada turno el usuario tendrá la posibilidad de controlar a uno de sus worms para moverlo y ejecutar ciertas acciones que dañarán al enemigo. El juego termina cuando uno de los ejércitos se quede sin worms.

4. Reglas

- Cada jugador tiene un ejército de 5 gusanos, la posición de inicio de cada uno de los gusanos es aleatoria dentro del mapa.
- El juego se divide por turnos. En cada turno el jugador correspondiente podrá controlar a uno de sus worms para caminar y realizar una acción. Luego de realizada la acción se acaba el turno y comienza el del oponente. Es importante notar que los gusanos van atacando en un cierto orden, es decir, primero se da la opción de acción al worm 1, luego al 2 y así sucesivamente.
- Una acción puede ser pasar el turno o atacar.
- Cuando un worm se queda sin HP debe desaparecer de la pantalla y ya no podrá seguir ayudando a su equipo.
- El juego termina cuando todos los worms de un ejército mueren, siendo el ganador el jugador que aún tiene worms vivos.

5. Funcionalidades

Para realizar el programa se piden las siguientes funcionalidades:

5.1. Funcionalidades mínimas

- Movimiento de worms: Al ser controlados por un jugador, los worms podrán moverse por el mapa. Cada turno el gusano que se va a mover debe tener un número limitado de movimientos de forma que el jugador no pueda posicionar su personaje inmediatamente al lado de un rival para luego disparar. Este límite de jugadas y cualquier otra característica del movimiento queda a su criterio mientras sea simple y coherente. Cualquier supuesto o decisión tomada en este punto mencionelo en el ReadMe. Tenga en cuenta que el terreno será irregular (ver más adelante características del terreno), por lo que dentro de este apartado tiene que haber una opción (tecla, click, boton en pantalla, etc) que permita al jugador saltar.
- Identificadores: Los worms tienen una cierta cantidad de vida, esta debe mostrarse gráficamente ya sea a través de números o una barra encima de la unidad. Junto con debe mostrarse el nombre del gusano el cual debe ser seleccionado aleatoriamente de una lista de al menos 20 nombres que usted debe definir. Como el juego consiste en dos equipos, tanto el HP como el nombre deben mostrarse con un texto del mismo color para todos los gusanos del mismo equipo.
- Ataques: Por cada turno el jugador tiene la opción de realizar o no un ataque. Cada ataque tiene un comportamiento específico que no discrimina entre equipos, es decir, un jugador que hace un ataque mal dirigido puede matar a sus compañeros de equipo. A continuación se detallarán los comportamientos y características, es importante notar que un arma puede tener más de un comportamiento asociado.
  - ☎ PROYECTIL
    Un arma de este tipo lanza un misil por los aires para llegar a su objetivo. Su trayectoria se ve afecta a la gravedad, por lo cual deberá tener un comportamiento parabólico.

[1] https://www.youtube.com/watch?v=u17ss8MXDCY

-  EXPLOSIVO:
  Un arma que causa una explosión, dañando a todo lo que se encuentra dentro del radio explosivo.
-  FÍSICO:
  Este tipo de armas solo puede ser utilizada a una distancia pequeña del oponente.
-  PERIÓDICO:
  Algunas armas dejan un efecto retardado en los worms afectados, causando daño periódico al comienzo de los siguientes turnos.
- DAÑO: Las armas causan variados niveles de daño y están clasificadas de la siguiente forma:
  -  Mortal: Suficiente daño para matar a un worm con HP completo.
  -  Alto: Hace un daño superior al 50 % del HP total de un worm.
  -  Moderado: Hace un daño entre el 25 % y el 50 % del HP total.
  -  Bajo: Hace un daño menor al 25 % del HP total.
  -  Insignificante: Daña menos del 5 % del HP total.
- RADIO: Algunas armas están sujetas a un efecto de área.
  -  Grande: Un círculo con el radio de aproximadamente tres worms.
  -  Moderado: un círculo con el radio de aproximadamente dos worms.
  -  Pequeño: un círculo con un radio de aproximadamente un worm.

  IMPORTANTE: Tenga en cuenta que los valores de los atributos son de referencia y usted podría modificarlos según le parezca conveniente. Al hacerlo, debe asegurarse que el comportamiento general se mantenga y que no se afecte negativamente la jugabilidad.

- Armamento: Cada unidad posee las siguientes armas:
  - Bazooka:    
    Esta es el arma principal. El radio de explosión y el daño son moderados.
  - Bat:  
    Esta arma sirve el propósito de golpear y humillar a tu oponente. Hace poco daño al enemigo, pero lo hace volar en la dirección que fue golpeado. La fuerza del golpe influye directamente en la distancia que deberá volar el worm bateado.
  - Air Strike:   
    Este es un ataque aéreo que daña varios lugares del mapa. El radio de explosión de cada uno de los misiles es menor que el de la bazooka. Queda a su criterio cómo se implementa la caída de los misiles. Este ataque se debe poder usar solo cada 4 turnos.

- Interacción: El usuario deberá ser capaz de elegir y realizar sus ataques con el mouse, para esto se le pide que implemente un sistema de drag and drop para lanzar los ataques. Es importante que este sistema muestre un feedback gráfico (MUY CLARO) en la interfaz para que el usuario sea capaz de ver en qué dirección y con cuánta fuerza lanzará su ataque (tipo angry birds, por ejemplo). Para el movimiento y la selección de armas usted puede implementar el teclado si así lo desea.
- Animación: Los distintos ataques deberán mostrarse en la interfaz con sus correspondientes animaciones. Esto incluye que los proyectiles se vean volando por los cielos y las explosiones y/o efectos de daño sean visibles.
- Datos históricos: Además de lo anterior, se deberá dar la posibilidad al usuario de ver los puntajes históricos. Estos toman en consideración todos las partidas jugadas anteriormente. Se dividen en tres categorías: daño hecho, unidades destruidas y ataques desperdiciados (que no dañan a ningún worm).

5.2. Funcionalidades extra

Usted deberá implementar al menos cuatro de las siguientes funcionalidades. Por cada funcionalidad extra que implemente correctamente se le dará un bonus de una décima hasta un tope de 5 décimas extra.

- Mapa dinámico:
  El terreno consistirá en una superficie dinámica que cambiará a medida que hayan ocurrido los distintos ataques. Esto quiere decir que cada vez que ocurra una explosión, se borrará un porción de la superficie correspondiente al radio de explosión (no necesariamente circular). Tome en consideración que esto afecta el terreno caminable por los worms.

- Bomba de tiempo: 
  Es un arma que se lanza como proyectil y que se adhiere al lugar del mapa donde cae. Luego de 2 turnos explota y causa un daño mayor que el de la bazooka.

- Muerte súbita: 
  Luego de unos cuantos turnos, el nivel del agua comienza a subir gradualmente. Si un worm toca el agua, este deberá morir ahogado.

- Sonido:
  Las armas y las acciones deberán tener feedback auditivo. En este lugar podrán encontrar buenos sonidos de referencia.

- Mapas aleatorios:
  Se deberá crear un mapa distinto para cada partida. Este deberá tener sentido y permitir las funcionalidades del juego. Se recomienda buscar imágenes y videos del juego original para ver qué reglas deberán considerar al momento de crear estos mapas de manera que no se perjudique el juego.

- Cámara dinámica:
  El mapa puede ser más grande de lo que la interfaz permite ver, por lo que la cámara mostrará solo una parte de este. Esta se enfocará en el worm que se encuentra jugando al comenzar cada turno, pero el usuario podrá moverla a su gusto.

- Editor de mapas:
  Permite crear el terreno en el que se jugará la partida siguiente. La manera recomendada de implementarlo es que el usuario pueda hacer click en distintos lugares de la interfaz y se deberán crear bloques en esas posiciones. Puede definir una manera distinta, pero debe ser clara y simple.

- Barriles explosivos: 
  En distintos lugares del terreno deberán existir barriles con gasolina. Estos explotarán si se encuentran en el rango de daño de un arma explosiva. El daño y radio de explosión son considerables.

- Arma biológica: 
  Es un nuevo prototipo de arma que los adrianinos diseñaron a partir de la información interceptada años atrás. Consta de un misil que explota al tocar el terreno y libera una nube de gases poco conocidos. Los worms en el rango de la explosión se ven afectados por algo que los humanos llaman cañitis alcoholicus. Esto no causa daño inicial, pero durante los siguientes 3 turnos, los afectados recibirán daño moderado.

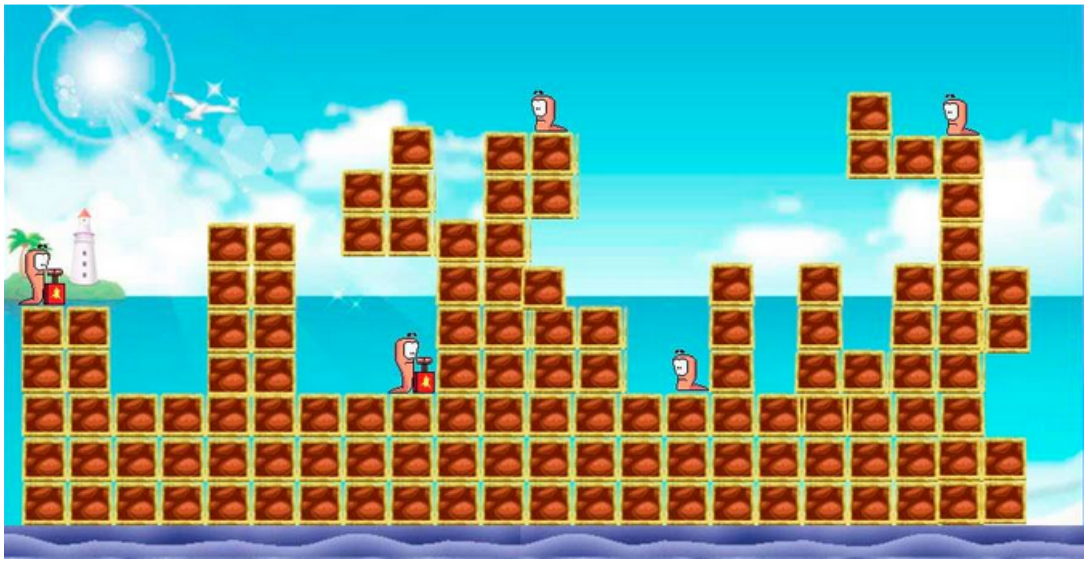
- Último suspiro: 
  Todos los worms terminarán su vida con honor. Para esto, al momento de morir detonarán una bomba causando una pequeña explosión alrededor suyo. Esta se comporta como arma explosiva.

- Teleportador:
  Este es un dispositivo de utilidad que se podrá utilizar solo una vez por cada jugador durante la partida. Permite cambiar de posición al worm actual a cualquier posición válida del mapa. El uso de este dispositivo no acaba el turno actual.

- Super Oveja: 
  Esta es un arma legendaria y por lo tanto no se puede describir con palabras. Para referencia observar este video. Solo se podrá utilizar una vez por juego y se deberá activar en un turno al azar, es decir, ninguno de los jugadores poseerá este arma en el primer turno. El daño acumulado no debe ser suficiente para matar a una unidad por sí solo.

6. Terreno de Juego y Unidades

A continuación se muestra un ejemplo muy básico de cómo es un terreno de juego pequeño. Este consiste en una grilla con bloques, cada bloque puede tener a lo más un worm. Si el bloque es destruido debe desaparecer ya no será una zona "pisable"por los gusanos. Si hay una torre en que la base es destruida, todos los bloques que están arriba de esta deben ser destruidos (no pueden haber bloques flotando). Tome en consideración que el terreno que usted diseñe debe ser capaz de albergar a 10 unidades, por lo que tendrá que crear un mapa bastante más grande que el que se muestra aquí.



7. Worms y armas

Para representar a los gusanos, las armas y las explosiones usted debe usar imagenes en formato .gif (para tener imágenes con animaciones) con el fin de que la interacción sea más dinámica. WPF nativamente no implementa la animación del gif, por lo que se le entregará una librería para poder manejarlos (ver siguiente sección). Usted es libre de eligir los gifs que desee siempre y cuando tengan coherencia y su interfaz no se vea afectada negativamente. Para las explosiones una web con muchas imágenes que le puede ser útil es la siguiente: http://www.yoda.arachsys.com/worms/wa/anims/index.html. Además se subirá junto con el enunciado un .zip con gifs de gusanos que le podrían ser útiles, sin embargo, su utilización es completamente voluntaria. En todo momento sientase libre de usar los gifs y la estética que quiera, las buenas interfaces serán premiadas.

8. Gif

Para realizar la tarea deberá usar sprites en formato gif, por lo que podrán hacer uso de la siguiente librería para poder usarlos:

- Uso en XAML:

  En la raíz de tu archivo XAML, declara el namespace:

```
<Window x:Class="WpfAnimatedGif.Demo.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:gif="http://wpfanimatedgif.codeplex.com">
```

  Luego, en `Image`, en lugar de poner `Source` como property, coloca `ImageBehavior.AnimatedSource` junto a la imagen en formato .gif:

```
    <Image gif:ImageBehavior.AnimatedSource="Images/animated.gif" />
```

  Si necesitas cambiar la forma en que se repite la animación, agrega la property ImageBehavior.RepeatBehavior:

```
        <!-- Repeat 3 times -->
        <Image gif:ImageBehavior.RepeatBehavior="3x"
               gif:ImageBehavior.AnimatedSource="Images/animated.gif" />

        <!-- Repeat for 10 seconds -->
        <Image gif:ImageBehavior.RepeatBehavior="0:0:10"
               gif:ImageBehavior.AnimatedSource="Images/animated.gif" />

        <!-- Repeat forever -->
        <Image gif:ImageBehavior.RepeatBehavior="Forever"
               gif:ImageBehavior.AnimatedSource="Images/animated.gif" />
```

  Los valores válidos para esta property incluyen:

  - Forever: La animación se repite por siempre.
  - Un número fijo de iteraciones.
  - Una duración de tiempo, especificada en formato de días, minutos, segundos, y fracciones de segundo.

- Uso en código:

  Para colocar una imagen utiliza el método `ImageBehavior.SetAnimatedSource:`

```
var image = new BitmapImage();
image.BeginInit();
image.UriSource = new Uri(fileName);
image.EndInit();
ImageBehavior.SetAnimatedSource(img, image);
```

  Puedes manejar las repeticiones de la animación con el método `ImageBehavior.SetRepeatBehavior:`

```
// Repetir 3 veces
ImageBehavior.SetRepeatBehavior(img, new RepeatBehavior(3));
// Repetir por 10 segundos
ImageBehavior.SetRepeatBehavior(img, new RepeatBehavior(TimeSpan.FromSeconds(10)));
// Repetir infinitamente
ImageBehavior.SetRepeatBehavior(img, RepeatBehavior.Forever);
```

- Control Manual de la animación:

  Primero para poder controlar la animación, debes obtener el `ImageAnimationController`:

  ```
  var controller = ImageBehavior.GetAnimationController(imageControl);
  ```

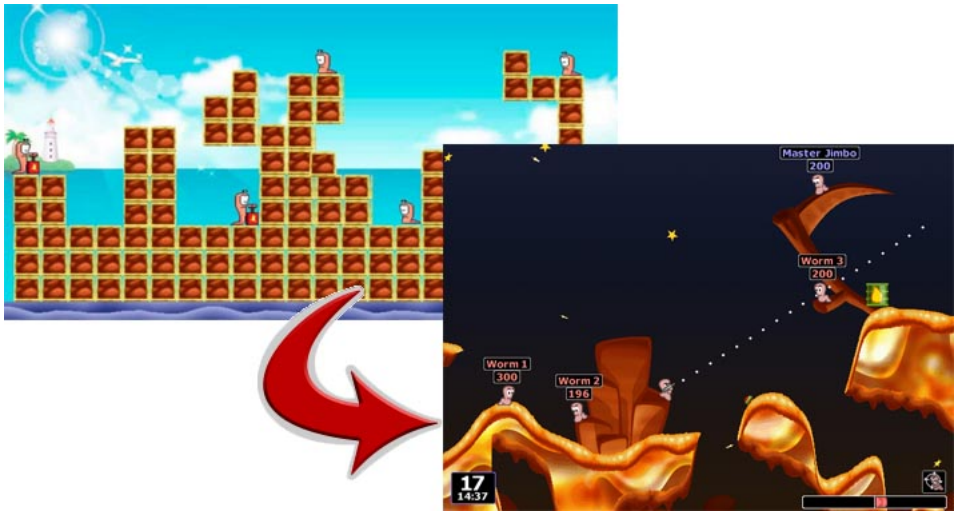
  Luego puedes usar para pausar, resumir o buscar:

  ```
  // Pausar la animacion
  controller.Pause();
  // Resumir la animacion(o reiniciarla si ya habia sido completada)
  controller.Play();
  // Ir al ultimo frame
  controller.GotoFrame(controller.FrameCount - 1);
  ```

9. Bonus

9.1. Superficies suaves (15 %)

Deberá implementar un mapa con superficies continuas y adecuar todas las funcionalidades implementadas para que operen correctamente con este nuevo terreno. (Este terreno se asemeja más al juego original).



Hint: piense en el manejo del mapa de fondo como un archivo para ver si hay suelo o cielo.

10. Restricciones y alcances

- Debido a las caracteristicas y la gran flexibilidad en el desarrollo de esta tarea, es imperativo que aclaren en el ReadMe las funcionalidades que decidieron implementar. De haber modificado alguno de los comportamientos o algo en la interfaz, explique ***brevemente*** las razones.
- Se debe realizar el programa con la lógica Back-End/Front-End. Está totalmente prohibido importar y utilizar las clases del Front-End en el Back-End.
- Tu programa debe ser desarrollado en el lenguaje C#, sobre .Net Framework 4.5.
- El entorno de desarrollo a utilizar debe ser Microsoft Visual Studio 2013.

- El ayudante puede bonificar o castigar tu puntaje[2] de la tarea, si le parece adecuado. Se recomienda ordenar el código.
- No subas las carpetas bin, obj ni el archivo .suo[3], hay un descuento de un punto en tu nota por hacerlo.
- Junto con tu tarea deberás subir un archivo ReadMe donde irá especificado todo lo que su programa hace o no, junto con las cosas que haya asumido. En caso de no subir este archivo se aplicará un descuento en su nota.

11. Entrega

- **Fecha/hora:** 30 de octubre de 2014 / 23:59.
- **Lugar:** En repositorios SVN personales, al interior del directorio `Tarea 4`.

Debe subir la solución de Visual Studio completa en código, sin ejecutables. Tareas que no compilen o no cumplan con las restricciones señaladas en este enunciado tendrán la calificación mínima (1.0).

[2] ±5 décimas.
[3] Es un archivo oculto.