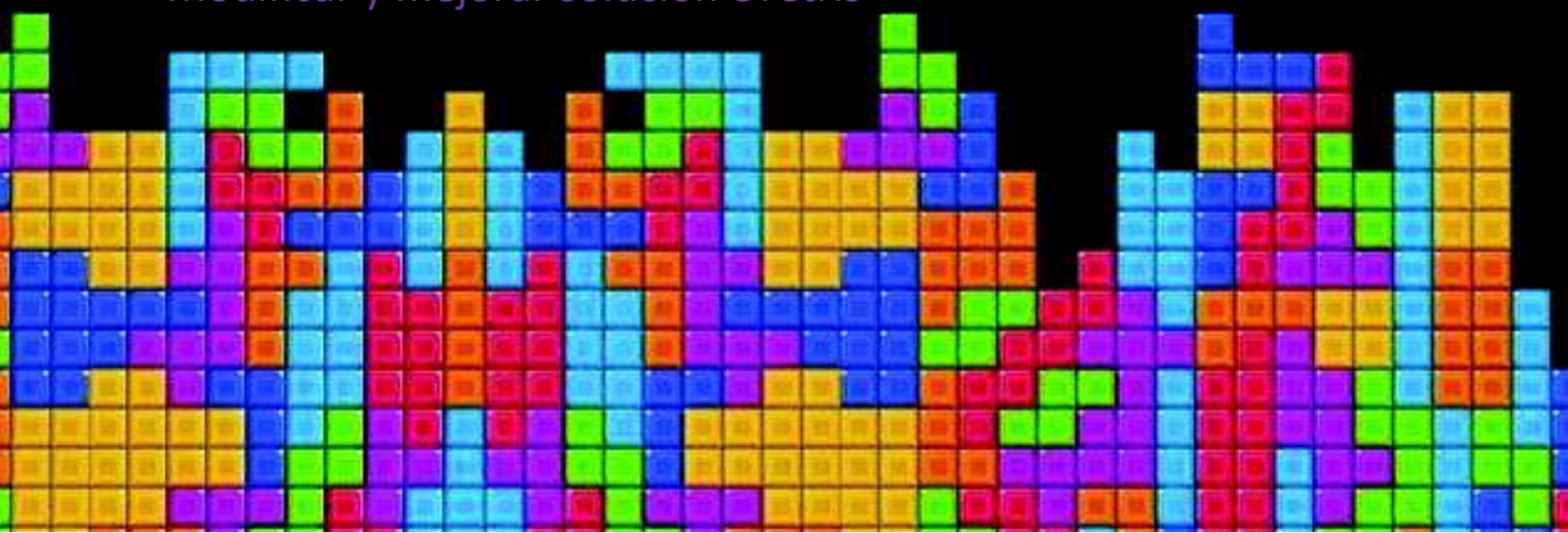
Rubén Arcos Ortega

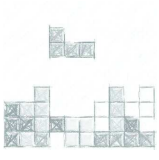
2º C.F.G.S. Desarrollo de aplicaciones multiplataforma

Servicios y procesos

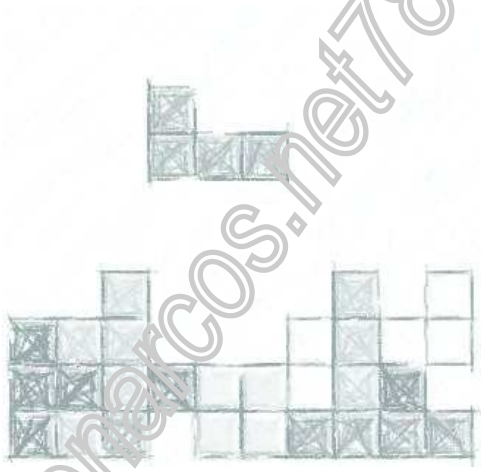
TETRIS

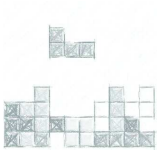
Modificar y mejorar solución eTetris





Índice

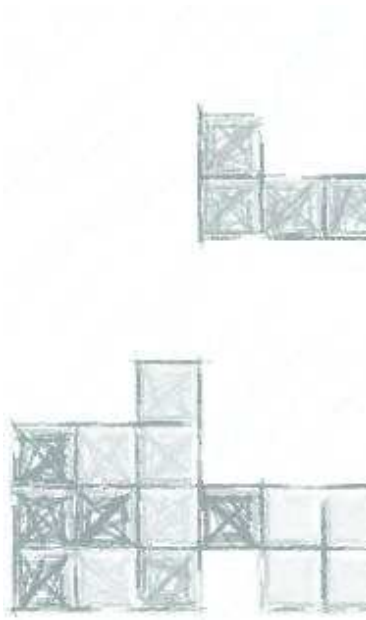


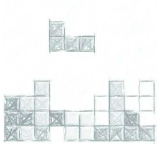


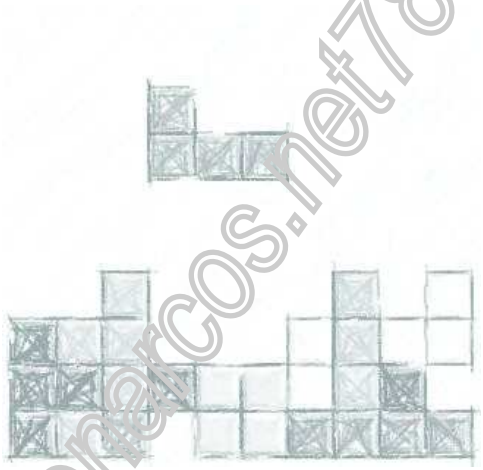
En la siguiente documentación, se detalla la práctica relacionada con la asignatura de Desarrollo de interfaces, con fecha de entrega del martes 01 de marzo de 2016.

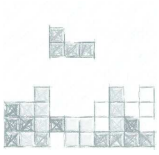
A continuación se encuentra el planteamiento del problema a resolver, la estructuración y motivación del proyecto, junto con los códigos fuentes relacionados.

He de dejar constancia que el proyecto ha supuesto una gran puesta en práctica de los conocimientos adquiridos durante el periodo de la asignatura en el ámbito de la programación mediante el lenguaje de programación C# en la plataforma .Net, dando paso al conocimiento del mismo y a una nueva visión de la programación orientada a objetos, como en los inconvenientes que con lleva este novedoso lenguaje de programación en continuo crecimiento.









1 Motivación del proyecto

Realización de mejoras y modificaciones de una aplicación facilitada en clase mediante el lenguaje de programación C#, que incorpore y requiera en la medida más cercana a la realidad del establecimiento del juego *Tetris*, con similitud a las normas y características del mismo. Utilización de entorno de desarrollo Visual Studio y la modularidad de la aplicación basado en patrón del MVC de la POO.

2 Planteamiento del problema real

2.1 Entorno

Sistema de control del juego *Tetris*, gestionando las partidas, control de movimientos a realizar, puntuaciones, utilización del registro del sistema y otros. Teniendo en cuenta la gestión de la interactuación con el usuario y los accesos a los datos de la partida.

2.2 Situación real y restricciones habituales

2.2.1 El juego

Consiste en la disposición de distintos *tetriminos*, figuras geométricas compuestas por cuatro bloques cuadrados unidos de forma ortogonal, las cuales se generan de una zona que ocupa 5x5 bloques en el área superior de la pantalla. Existen distintas versiones del juego. La original tiene siete piezas diferentes y en todas las versiones se tratan de representaciones de letras del alfabeto occidental o cirílico. No hay consenso en cuanto a las dimensiones para el área del juego, variando en cada versión, pero esta es la utilizada en el caso que nos ocupa. Sin embargo, dos filas de más arriba están ocultas al jugador.

2.2.2 Piezas del Tetris

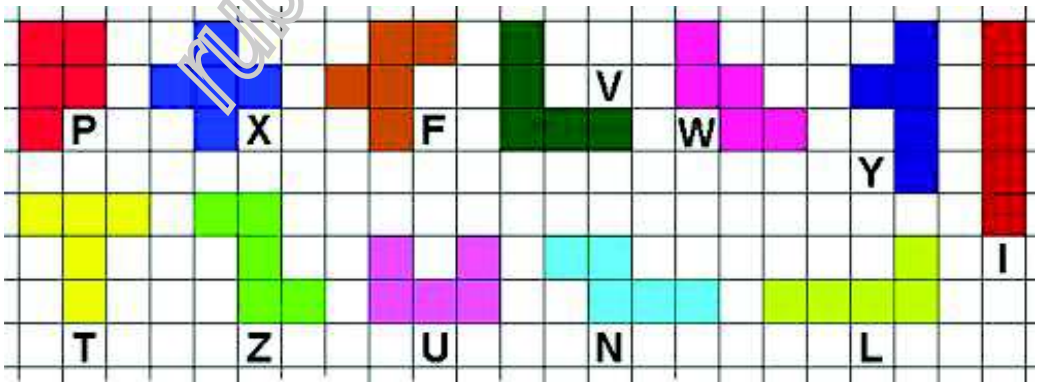
El jugador no puede impedir esta caída, pero puede decidir la rotación de la pieza (0°, 90°, 180°, 270°) y en qué lugar debe caer. Cuando una línea horizontal se completa, esa línea desaparece y todas las piezas que están por encima descienden una posición, liberando espacio de juego y por tanto facilitando la tarea de situar nuevas piezas. La caída de las piezas se acelera progresivamente. El juego acaba cuando las piezas se amontonan hasta llegar a lo más alto (3x5 bloques en el área visible o lo indicado por el usuario, puesto que en nuestro caso ha sido implementado de forma dinámica), interfiriendo la creación de más piezas y finalizando el juego.

2.2.3 Los jugadores

El número de jugadores es solamente de uno en cada jugada. No obstante, cada jugada será almacenada junto con las iniciales del jugador. Se permite la realización de varias partidas con las mismas iniciales, para que el jugador pueda batir sus propios records.

2.3 Restricciones y consideraciones aplicadas en la aplicación

- Se han incluido doce piezas (*tetriminos*) diferentes. Quedando así las formas y colores:



- Todas las piezas permiten cuatro rotaciones diferentes hasta alcanzar los 360º, excepto la letra “P” que se ha bloqueado su rotación.
- Todos los niveles requieren de la realización de dos líneas para superarse.

- Los niveles se distinguen en la velocidad de movimiento de la pieza, el tiempo de espera en avanzar un línea de la matriz.
- La puntuación constará de cuatro puntos por cada pieza establecida en una posición final. Se incrementará sin límite alguno.
- La puntuación obtenida será almacenada en el momento de finalizar la partida, o el cierre de la aplicación.
- Se requiere de forma obligatoria la inclusión de las iniciales del jugador, para almacenar su puntuación en el ranking general.
- La aplicación almacena la última localización de la ventana en pantalla.
- La puntuación podrá consultarse cuando no haya una partida en curso o en el estado de pausa.
- Se podrá iniciar una nueva partida en cualquier momento teniendo en cuenta que se perderán los puntos de la última partida en curso.
- El juego puede pausarse en cualquier momento, y la utilización de otra funcionalidad del mismo, implica la activación y desactivación de la pausa.
- El número de filas y columnas de la matriz de juego, podrá ser ampliada o reducida al comienzo de la aplicación y será mantenida esta característica durante todo el tiempo de uso de la misma.

3 Análisis y diseño del problema

3.1 Estructura y lógica de la aplicación

3.1.1 Clase principal [Juego.cs]

- Contiene el inicio del flujo de ejecución de la aplicación.
- Gestiona la comprobación de la resolución de pantalla.
- Lanzamiento de la ventana de splash y posteriormente la interfaz gráfica de la aplicación.

3.1.2 Formulario de splash `[frmSplash.cs]`

- Muestra durante cinco segundos la información de la aplicación.

3.1.3 Clase de interfaz gráfica, interacción de usuario y menús `[frmGUI.cs]`

Debido a la cantidad de métodos y funcionalidades a tener en cuenta para esta clase, he realizado la división por regiones, con el motivo de la facilidad de compresión.

- `[Controles]`: Gestiona la detección de las teclas de los cursores del teclado y su inicialización.
- `[Dibujo piezas]`: Calcula las dimensiones de la matriz, con respecto a las especificaciones dadas y dibuja la siguiente pieza que mostrará el juego.
- `[Lógica juego]`: Detecta el número de filas realizadas, el movimiento de las piezas y cuando todas las filas han alcanzado la última fila de la parte superior. También los requisitos para subir de nivel (2 líneas) y el registro de la puntuación. Por último, los estados del juego: nuevo, pausa y reinicio; con las acciones correspondientes.
- `[Menús y eventos]`: Todo lo relacionado con la visualización y acciones de los menús, elementos de interacción y cambios de estado de la aplicación.
- `[Registro]`: Las operaciones con el registro del sistema: carga, grabación y comprobación de los elementos del registro de la aplicación: puntuaciones y posición de la ventana.

3.1.4 Clase con la estructura y elementos del juego `[Tetris.cs]`

- Gestiona todas las piezas que se dispondrán en el juego, con las características de: forma, rotación y color.
- Gestión de las propiedades del juego: matriz (zona de movimiento de las piezas), estado del juego, pieza siguiente, puntuación, líneas del jugador, nivel en curso, los movimientos y rotaciones de la pieza actual, la inicialización del juego y del nivel, la inicialización del juego completo y la detección de las líneas realizadas.

3.1.5 Clase de cada pieza [Pieza.cs]

- Contiene todas las propiedades y funcionalidades de cada pieza: forma, color, rotación, posición actual y tamaño.
- Dispone de la sobreescritura de la clonación del objeto pieza para que genere una nueva pieza diferente con cada instancia de esta.

3.1.6 Clase de la zona de juego [Matriz.cs]

- Contiene las propiedades de la zona de movimiento de piezas, la matriz de la pantalla, con las dimensiones basadas en las filas y columnas de esta.
- Gestiona las colisiones y movimientos disponibles para/entre las piezas.
- Refresco del dibujo y pintado de la pantalla, en este caso la zona de la matriz.
- Acción predeterminada en caso de realizar línea.

3.1.7 Clase de control de la pieza [FormaPieza.cs]

- Gestiona el área ocupada por la pieza, y las dimensiones totales de la misma.

3.1.8 Constantes globales [Constantes.cs]

- Contiene todas las constantes de ámbito público del juego y los listados de colores y tiempos por nivel.

3.1.9 Utilidades y recursos [Utiles.cs, GestionRegistro.cs]

- [Utiles.cs]: Gestiona la comprobación de la resolución de pantalla.
- [GestionRegistrpo.cs]: Gestiona las operaciones comunes con el registro del sistema.

3.1.10 Formulario acerca de y puntuaciones [AcercaDe.cs, frmAcercaDe.cs y frmPuntuaciones.cs]

- [AcercaDe.cs]: Formulario acerca de del autor del juego.
- [frmAcercaDe.cs]: Formulario del autor de las modificaciones realizadas.
- [frmPuntuaciones.cs]: Gestiona la visualización del histórico de puntuaciones del juego, ordenados por los puntos.

3.2 Diagrama de clases y estructura de la aplicación

Debido a las características de la aplicación, he considerado la mejora forma de resolverlas centrándome en la estructura de la creación de las piezas y el flujo del juego desde el comienzo, diferenciando tres fases:

- Creación e inicialización del juego.
- Generación de las piezas.
- Detección de eventos y acciones en el juego.

La creación e inicialización del juego se ejecuta:

- [Juego]: Detecta que el sistema cumpla con la resolución mínima recomendada.
- [frmGUI]: Crea una instancia de `Tetris`, el cual crea una instancia de `MatrizPantalla` y `Pieza` tantas como se han generado en la clase `Tetris`.

La generación de piezas realiza:

- [Pieza]: Crea cuatro `FormaPieza` según rotación, y las propiedades de cada pieza. Esta es creada desde la clase `Tetris` por cada una de las doce piezas definidas.
- [FormaPieza]: las rotaciones de la pieza y sus propiedades correspondientes.
- [MatrizPantalla]: tamaño de la matriz, sus propiedades, detección de colisiones, y refrescos de dibujado de pieza.

Detecciones de eventos y acciones del juego:

- [frmGUI]: Realiza las llamadas a `MatrizPantalla` y `Tetris` para la realización de los movimientos de las piezas, las detecciones de línea, las detecciones de colisiones, los refrescos de pantalla y la generación de nuevas piezas aleatorias. También gestiona los estados del juego y la música del mismo.

4 Programación fuente

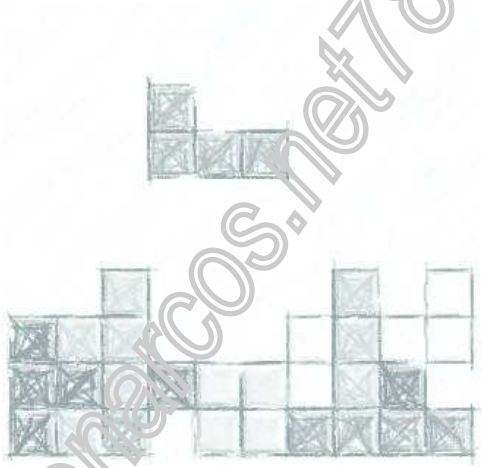
Se adjuntan, como solución de proyecto para Visual Studio 2010 – 2015 (Ed. Comunity) y .Net Framework 4.0.

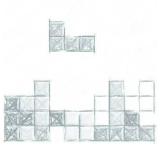
5 Justificación de la solución

Del análisis realizado en apartados anteriores acerca de la problemática que nos ocupa así como de los objetivos a lograr, se desprende que las necesidades de la gestión del juego *Tetris* son muy específicas, por lo que la aplicación actual no satisface dichas necesidades en su totalidad, al tener que ser ajustada para la solución dada.

Es necesario, por tanto, recurrir al diseño personalizado de la aplicación, utilizando para ello un entorno de desarrollo adecuado que facilite la creación de la misma. En este caso particular se ha reducido el número de condicionantes a tener en cuenta para la realización de las puntuaciones, niveles y fases de interacción con el jugador para poder gestionarse de la forma más veraz posible mediante la programación con técnicas orientadas a objetos en C#. La utilización de un objeto que contenga la lógica de la partida se ha contemplado como un objeto independiente para mantener la coherencia en el nivel de abstracción. Este es el mismo hecho por el cual se han separado los elementos puntuaciones y registro del sistema para mantener el patrón de diseño lo más cercano al modelo-vista-controlador. El resto de elemento se ha incorporado a las clases existentes por su universalidad y limitación de la funcionalidad.

Otro aspecto a tener en cuenta es la estructura en la que se encuentra el proyecto, la cual está pensada en la posterior escalabilidad del mismo, la reutilización de objetos y la incorporación de nuevas funcionalidades, intentando por este hecho disponer de la mayor parte de elementos dinámicos, como el uso de la ampliación de las constantes e instanciación de objetos desde el punto de inicio de la aplicación en proyectos independientes.





6 Bibliografía

- Documentación propia y facilitada en clase.
- https://es.wikipedia.org/wiki/Tetris