Computer programming

Final Project

Report

I.E.S. San Vicente
San Vicente del Raspeig (Alicante)
2014/2015

Teacher:
José Ignacio Cabanes

Student:
Luis Miguel Rubio Toledo

1. Introduction

Project name

- Tails

Made by
Luis Miguel Rubio Toledo

Short description of the project
Is the typical game arcade “sonic” but only play with the protagonist “Tails” (friend's sonic), a slightly altered, instead of his colour, he will be black colour, and he will move from left to right in the screen to end the level. I will use a graphical application with SDL graphics library, created by the teacher.

2. Functionality of the project

After entering the program, a welcome screen will we displayed, where the user can choose between:

- Play.
- See score the best players.
- See help for play.
- See Credits.
- Quit the game.

Screen Play:
The player will use to Tails for the purpose of complete this level. Tails can move to left, to right, jump, he can do a curl to kill the enemies or can to slide.
He can run and take items (rings).
If Tails has any ring and hit him then he lost all rings.
If take 100 ring before hit him then, he will create a life, if he has not any ring then he lost a life, and he will start a new this level.
Tails start the game with 3 lifes.
If lives of Tails are 0 then finish the party, and you must put your name, if your score are into the 10 best it will save in a file.

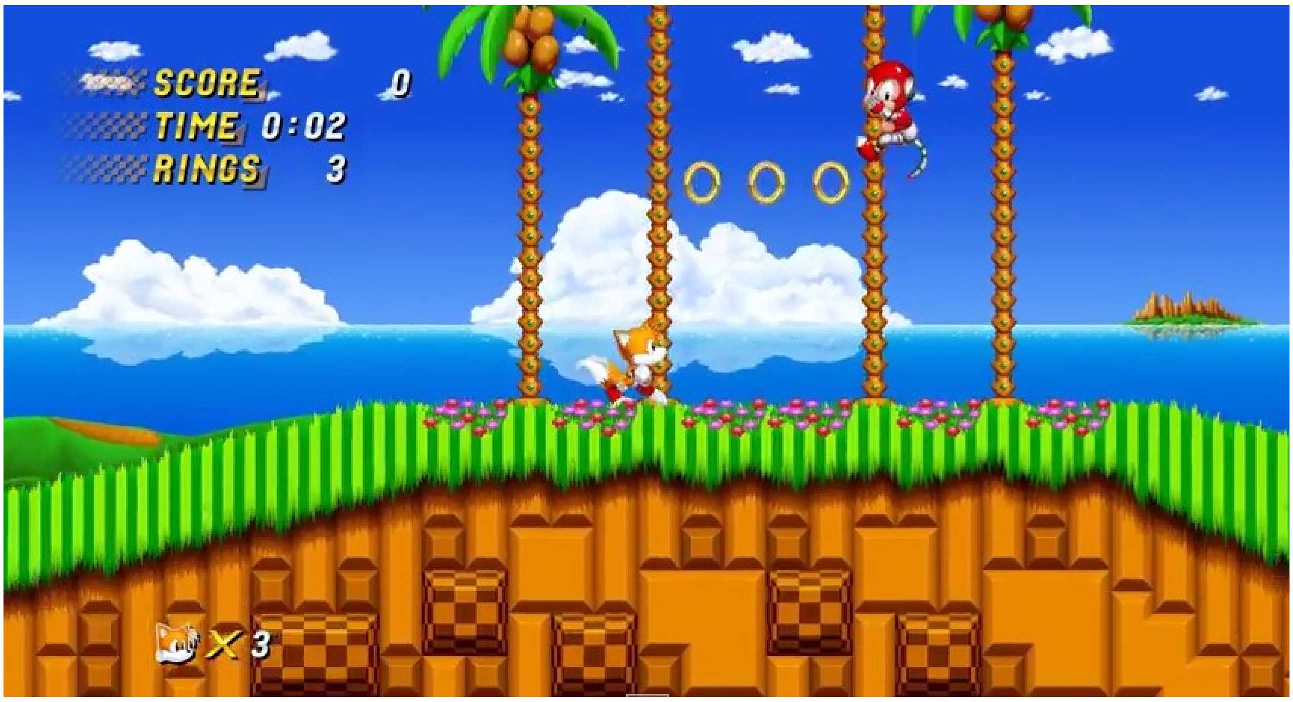
Screen Score:
it will see 10 best scores and use Esc for to go back.

Screen Help:
it will see all keys to use.

Screen Credits:
it will display data about the programmer

3. Screen prototype

The game screen will look like this:



4.Analysis

4a. Requisites

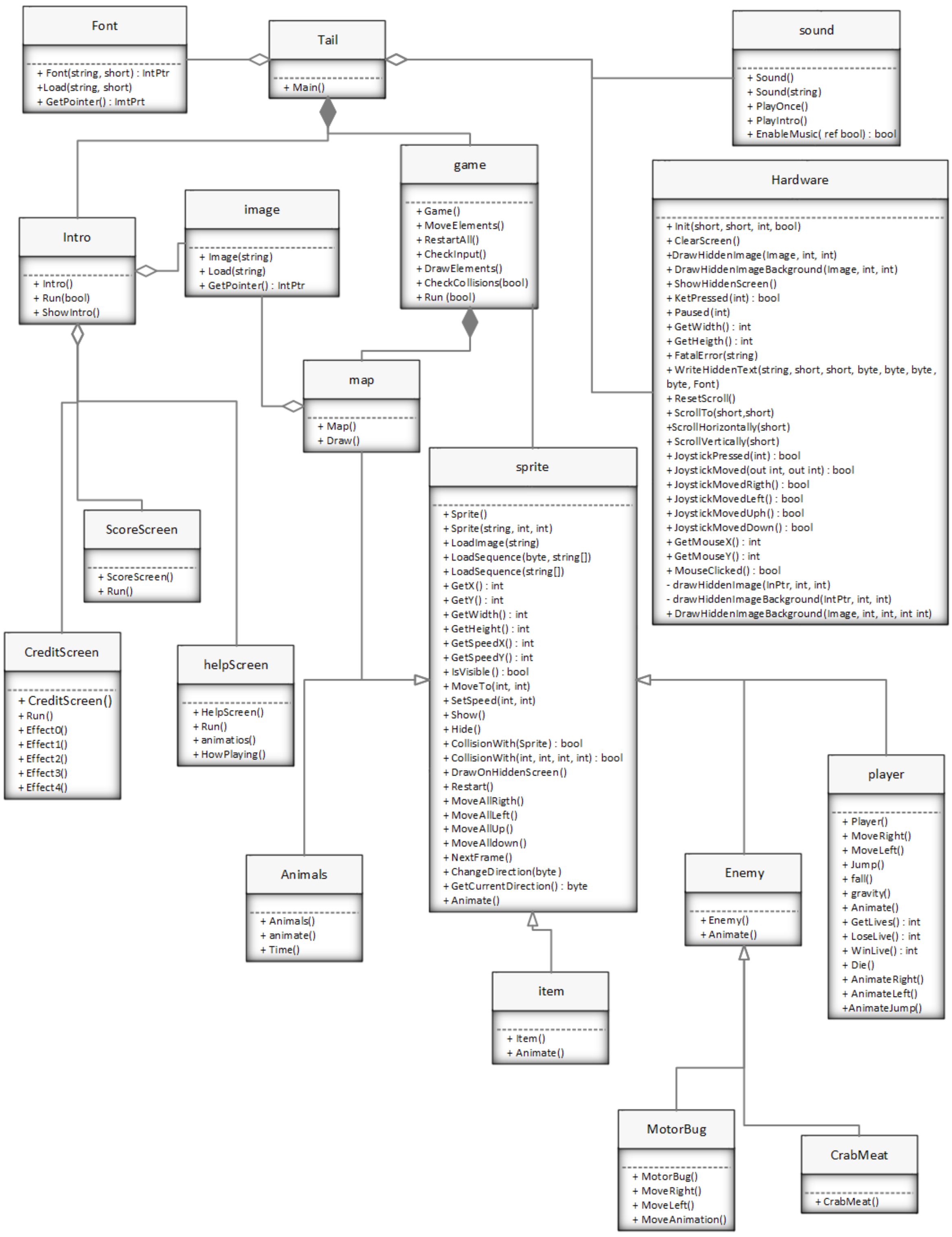
Requisite	Date achieved
The game will start showing an intro screen	Done
The intro screen will allow he user to enter a Help screen, which will display hints on how to play	Done
The intro screen will allow he user to enter a Credits screen, which will display data about the programmer	Done
The game can be paused pressing "P", and then returned by pressing any other key	To Do
Tails will can move right, left, jump, and slide correctly	left and right
Tails must take the ring	Done
Tails must see with movement with sprite	Done
The intro screen will allow he user to enter a Score screen, which will display High scores	Done
When the program are in play, must display background map, score, life	Done
The player will must see the enemies, the items, and Tails	Done
The player will listen the music when start play	Done
The player will listen the sounds when Tails jump, or slide	To Do
Tails must kill the enemies correctly	Done
Tails must catch the rings correctly	Done
(...)	

4b. Basic pseudocode

```
Program Main Body:
    initialize size screen
    initialize intro
    create bool gameStatus = false
    do
      run intro
      if exist intro.gamechosen then
        initialize game
        run game
      end if
      if press key escape then
        gameStatus = true
      end if
    while not is gameStatus true
End program Main Body
```

(...)

4c. Classes diagram



5. Initial planning and expected deliveries

5a. Expected deliveries

1. create class Hardware, using Tao.SDL
2. After the Intro screen, create Class Sprite, class player and class enemy
3. create class help
4. create other class (only for “To Do”)
5. move with using the arrow keys left and right (only player)
6. testing movements of player correctly
7. must enter and exit the options on IntroScreen correctly
8. draw enemies (minimun 2)
9. create class items (for rings) and draw
10. create method to colisions in class Game
11. colisions with items
12. colisions with enemies
13. create map class and draw
14. implement movement with background(map)
15. implement score, life, number caught ring in game
16. create movement sprite to Tails
17. implement names in class ScoreScreen
18. the enemies must move
19. create movement sprite animation to ring
20. must Restart game
21. Collisions with background
22. inplement other item (box)
23. create movement sprite animation to the enemies
24. implement name in class credits and draw
25. create method to gravity (jump)
26. create method for effect to quick
27. create method for use the curl
28. implement colisions only when Tails is in mode curl
29. use “p” for pause the game
30. save the game in a text file
31. implement sound for main screen (Intro Screen)
32. implement sound when jump
33. implement sound when use the curl
34.

(…) Note: You must plan **33 deliveries**. Each one of them will correspond to 1 hour of work in the classroom.

5b. Real deliveries

```
0.01 06-03-2015: create class Hardware, Font based on SdlMuncher 0.14, class Init
(Initial version).
0.02 06-03-2013: create class Image Initial version.
0.03 27-03-2015: create class Sprite, class player.
0.04 22-05-2015: reestruct classes Tails (now is Main) and Intro, create class Game.
The game can be quit (ESC) or start (SPC) correctly.
0.05  28-05-2015: show player and enemies
0.06  02-05-2015: move with using the arrow keys left and right
0.07  03-05-2015: show 20 rings consecutives
0.08  02-06-2015: check colision with rings
0.09  02-06-2015: check colision with enemies
0.10  04-06-2015: show map
0.11  04-06-2015: move rings with map
0.12  04-06-2015: show CreditScreen
0.13  04-06-2015: show ScoreScreen
0.14  04-06-2015: implement sequence sprites in class sprite
0.15  05-06-2015: create movement sprite animation to ring
0.16  05-06-2015: control of time in movements of sprites with DataTime
0.17  05-06-2015: show menu with all cases
0.18  05-06-2015: create swich with 4 effects in class creditScreen
0.19  05-06-2015: create movement sprite animation
0.20  05-06-2015: the enemies have movement in game
0.21  05-06-2015: Restart game correctly
0.22  05-06-2015: create gravity
0.23  06-06-2015: check colision with sprites
0.24  06-06-2015: create struct for CurrentScore
0.25  28-06-2015: create menu and method HowPlaying in HelpScreen
0.26  28-06-2015: create animation in HelpScreen
0.27  06-06-2015: implement methods
0.28  06-06-2015: implement sound
0.29  06-06-2015: add life when take 10 rings correctly
0.30  06-06-2015: clean and debug in collisions
0.31  06-06-2015: implement arraList and load file in class ScoreScreen
0.32  07-06-2015: display all screen background
0.33  07-06-2015: show animals when enemy death
```

6. File formats

6a. Plain files format

The scores table is saved to a text file, which contains the name of a player and the corresponding score in one line, and the following line other player, and so on, to complete the 10 highest scores, as in this example:

01000 - Pepe
00500 - Juan
02000 - Javi
00060 - Alvaro
02600 - Laura
02500 - Mimi

7. Problems found and solutions

I have problems in jump, I am not implement correctly, and now Tails fly not jump.

Other problem are collisions player with enemy when Tails jump, because he not must death only must kill him, finally I implement correctly.

The map had not scroll, not print all map, after several attempts I created an image more big than screen and he can move all map.

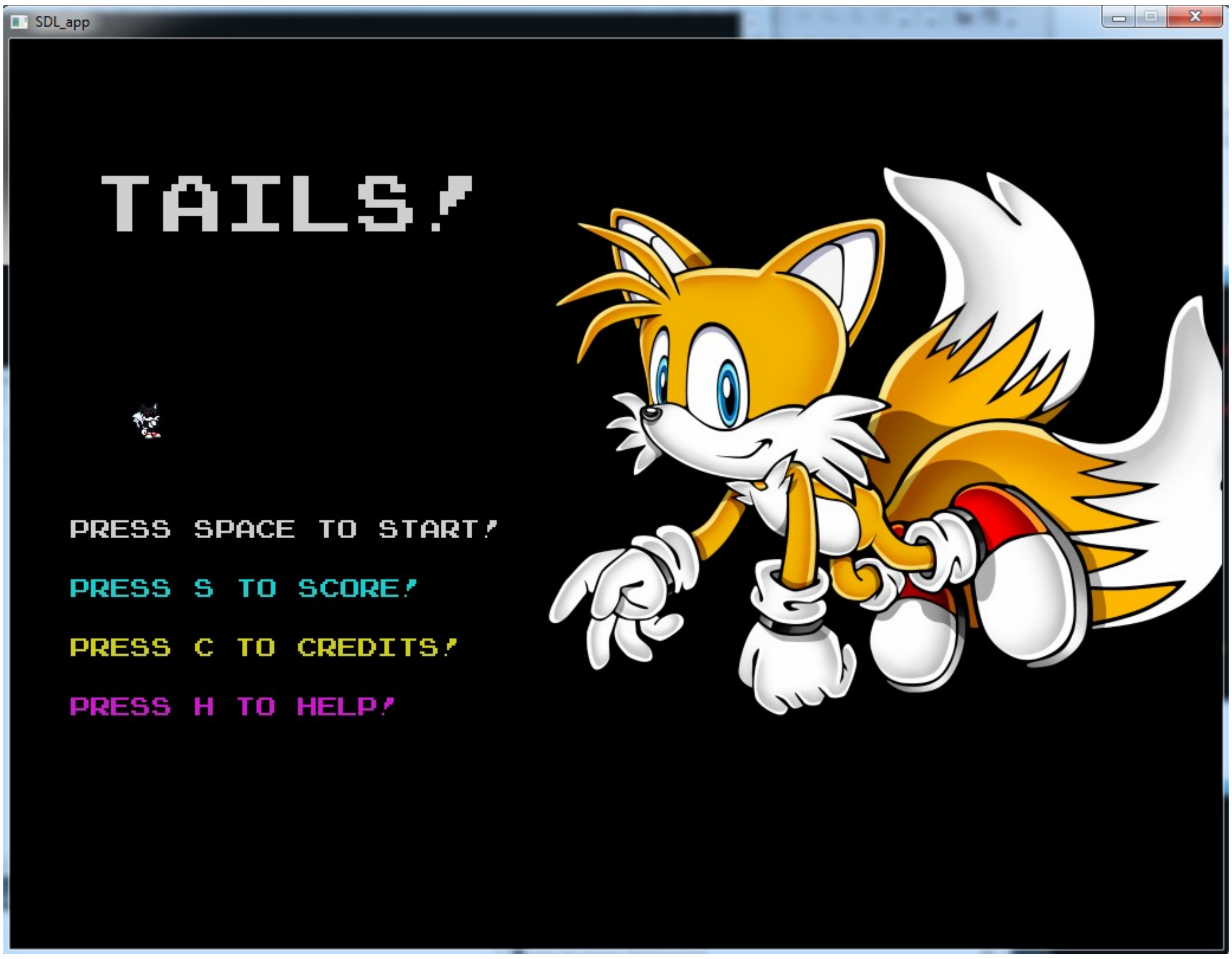
(…)

8. Improvements or restrictions to the starting design

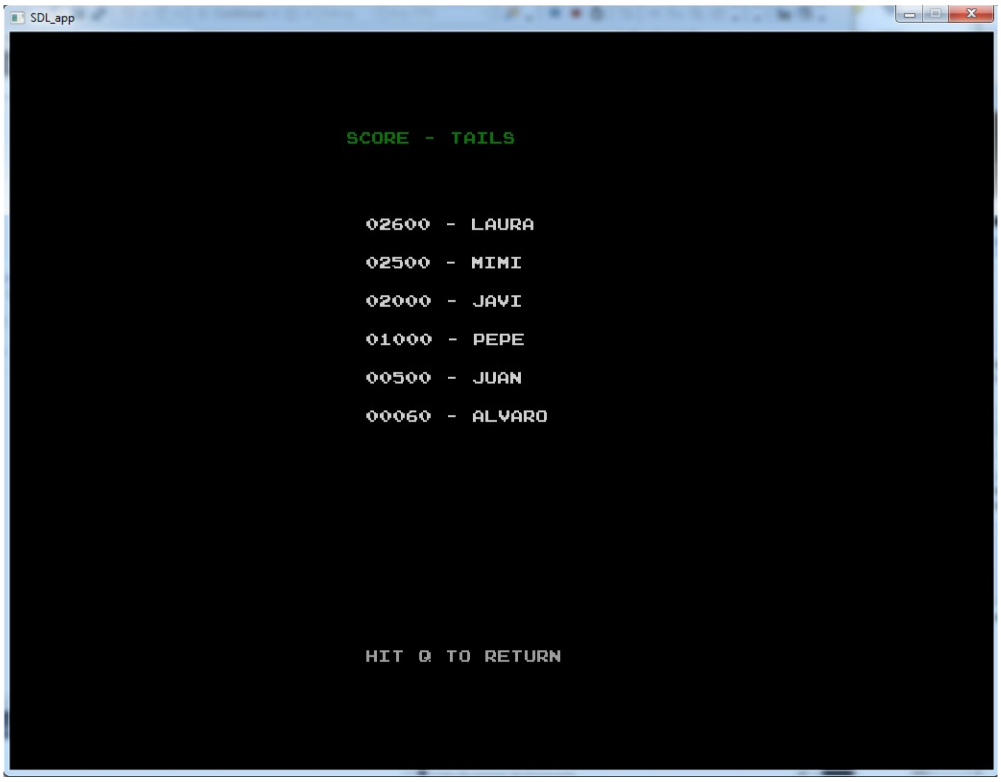
(…)

9. Screenshots of the final project

- Intro Screen



- Score Screen



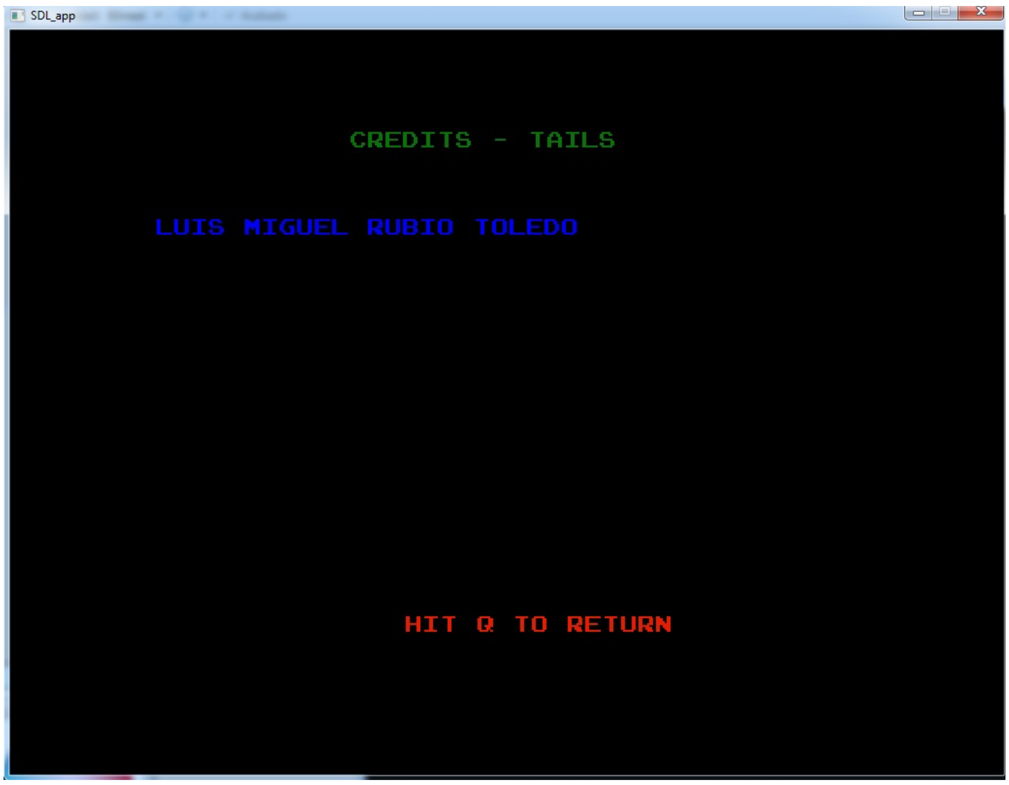
SDL_app

```
SCORE - TAILS

02600 - LAURA
02500 - MIMI
02000 - JAVI
01000 - PEPE
00500 - JUAN
00060 - ALVARO

HIT Q TO RETURN
```

- Credits Screen



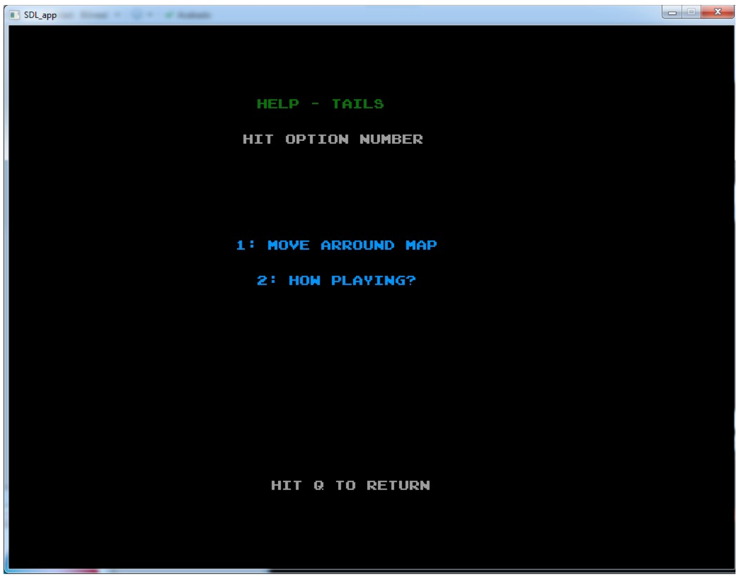
SDL_app

```
CREDITS - TAILS

LUIS MIGUEL RUBIO TOLEDO

HIT Q TO RETURN
```

- Help Screen



SDL_app

HELP - TAILS

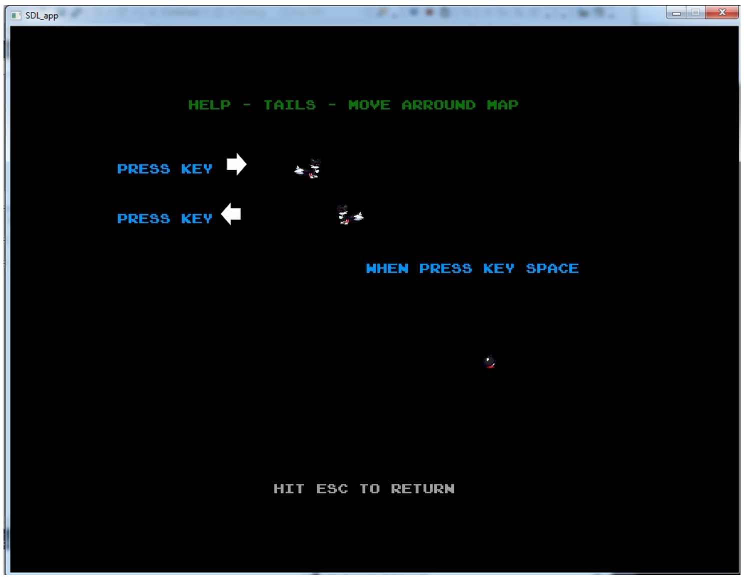
HIT OPTION NUMBER

1: MOVE ARROUND MAP

2: HOW PLAYING?

HIT Q TO RETURN

- options in help Screen



SDL_app

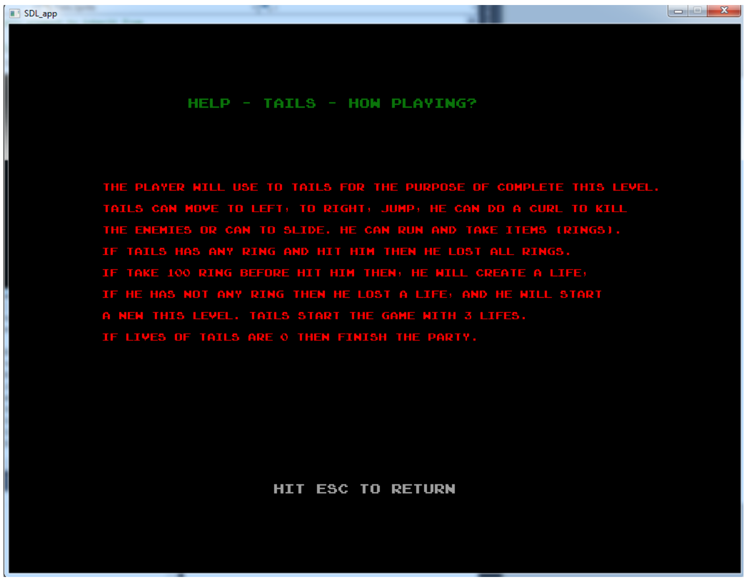
HELP - TAILS - MOVE ARROUND MAP

PRESS KEY ➡

PRESS KEY ⬅

WHEN PRESS KEY SPACE

HIT ESC TO RETURN



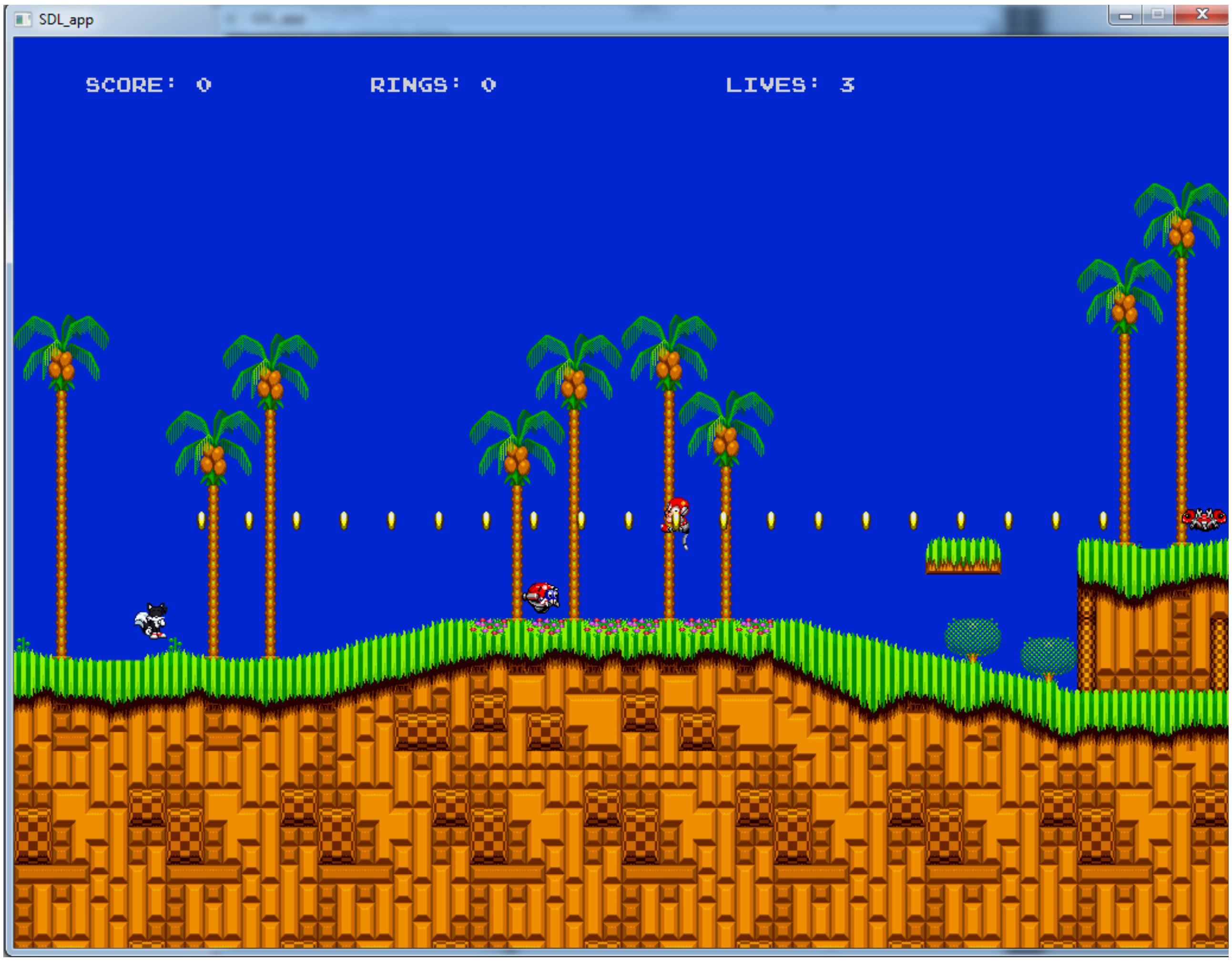
SDL_app

HELP - TAILS - HOW PLAYING?

THE PLAYER WILL USE TO TAILS FOR THE PURPOSE OF COMPLETE THIS LEVEL.
TAILS CAN MOVE TO LEFT, TO RIGHT, JUMP, HE CAN DO A CURL TO KILL
THE ENEMIES OR CAN TO SLIDE. HE CAN RUN AND TAKE ITEMS (RINGS).
IF TAILS HAS ANY RING AND HIT HIM THEN HE LOST ALL RINGS.
IF TAKE 100 RING BEFORE HIT HIM THEN, HE WILL CREATE A LIFE,
IF HE HAS NOT ANY RING THEN HE LOST A LIFE, AND HE WILL START
A NEW THIS LEVEL. TAILS START THE GAME WITH 3 LIFES.
IF LIVES OF TAILS ARE 0 THEN FINISH THE PARTY.

HIT ESC TO RETURN

- Game Start Screen



(...)

10. Source code of the final project

```
/**
 * Tails.cs - Partial sonic clone
 *
 * Luis Miguel Rubio Toledo, 2015
 *
 * Changes:
 * 0.01, 06-03-2015: Initial version, based on SdlMuncher 0.14
 * 0.02, 06-03-2015: Show Intro...
 * 0.03, 27-03-2015:  create class Sprite and Player
 * 0.04, 22-05-2015: change class name "program" for "Tails"
 *                   reestruct the class Tails, only Main
 *

 */
using System;


namespace Tails
{
    class Tails
    {
        /// <summary>
        /// Execute game
        /// </summary>
        public static void Main()
        {
            bool onMusic = true;
            bool fullScreen = false;
            Hardware.Init(1024, 768, 24, fullScreen);

            Intro myIntro = new Intro();


            bool gameStatus = false;
            // Game Loop
            do
            {
                myIntro.Run(onMusic);

                if (myIntro.GameChosen)
                {
                    Game g = new Game();
                    g.Run(onMusic);
                }

                if (Hardware.KeyPressed(Hardware.KEY_ESC))
                    gameStatus = true;


            }
            while (!gameStatus);
        }
    }
}


/**
 * SdlHardware.cs - Hides SDL library
 *
 * Luis Miguel Rubio Toledo, 2015
 *
 * Changes:
 * 0.01, 06-03-2015: Initial version, based on SdlMuncher 0.14
 * 0.32, 07-06-2015: implement new method for background
 */
using System.IO;
using System.Threading;
using Tao.Sdl;
using System;

namespace Tails
{



    class Hardware
    {
        static IntPtr hiddenScreen;
        static short width, height;

        static short startX, startY; // For Scroll
        static bool isThereJoystick;
        static IntPtr joystick;

        static int mouseClickLapse;
        static int lastMouseClick;


        public static void Init(short w, short h, int colors, bool fullScreen)
        {
            width = w;
            height = h;

            int flags = Sdl.SDL_HWSURFACE | Sdl.SDL_DOUBLEBUF | Sdl.SDL_ANYFORMAT;
            if (fullScreen)
                flags |= Sdl.SDL_FULLSCREEN;
            Sdl.SDL_Init(Sdl.SDL_INIT_EVERYTHING);
            hiddenScreen = Sdl.SDL_SetVideoMode(
                width,
                height,
                colors,
                flags);

            Sdl.SDL_Rect rect2 =
                new Sdl.SDL_Rect(0, 0, (short)width, (short)height);
            Sdl.SDL_SetClipRect(hiddenScreen, ref rect2);

            SdlTtf.TTF_Init();

            SdlMixer.Mix_OpenAudio(22050, unchecked(Sdl.AUDIO_S16LSB), 2, 1024);

            // Joystick initialization
            isThereJoystick = true;
            if (Sdl.SDL_NumJoysticks() < 1)
                isThereJoystick = false;

            if (isThereJoystick)
            {
                joystick = Sdl.SDL_JoystickOpen(0);
                if (joystick == IntPtr.Zero)
                    isThereJoystick = false;
            }

            // Time lapse between two consecutive mouse clicks,
            // so that they are not too near
            mouseClickLapse = 10;
            lastMouseClick = Sdl.SDL_GetTicks();
        }

        public static void ClearScreen()
        {
            Sdl.SDL_Rect origin = new Sdl.SDL_Rect(0, 0, width, height);
            Sdl.SDL_FillRect(hiddenScreen, ref origin, 0);
        }

        public static void DrawHiddenImage(Image image, int x, int y)
        {
            drawHiddenImage(image.GetPointer(), x + startX, y + startY);
        }

        public static void DrawHiddenImageBackground(Image image, int x, int y)
        {
            drawHiddenImageBackground(image.GetPointer(), x + startX, y + startY);
        }

        public static void ShowHiddenScreen()
        {
            Sdl.SDL_Flip(hiddenScreen);
        }

        public static bool KeyPressed(int c)
        {
            bool pressed = false;
            Sdl.SDL_PumpEvents();
            Sdl.SDL_Event myEvent;
            Sdl.SDL_PollEvent(out myEvent);
            int numkeys;
            byte[] keys = Tao.Sdl.Sdl.SDL_GetKeyState(out numkeys);
            if (keys[c] == 1)
                pressed = true;
            return pressed;
        }

        public static void Pause(int milisegundos)
        {
            Thread.Sleep(milisegundos);
        }

        public static int GetWidth()
        {
            return width;
        }

        public static int GetHeight()
        {
            return height;
        }

        public static void FatalError(string text)
        {
            StreamWriter sw = File.AppendText("errors.log");
            sw.WriteLine(text);
            sw.Close();
            Console.WriteLine(text);
            Environment.Exit(1);
        }

        public static void WriteHiddenText(string txt,
            short x, short y, byte r, byte g, byte b, Font f)
        {
            Sdl.SDL_Color color = new Sdl.SDL_Color(r, g, b);
            IntPtr textoComoImagen = SdlTtf.TTF_RenderText_Solid(
                f.GetPointer(), txt, color);
            if (textoComoImagen == IntPtr.Zero)
                Environment.Exit(5);

            Sdl.SDL_Rect origen = new Sdl.SDL_Rect(0, 0, width, height);
            Sdl.SDL_Rect dest = new Sdl.SDL_Rect(
                (short)(x + startX), (short)(y + startY),
                width, height);

            Sdl.SDL_BlitSurface(textoComoImagen, ref origen,
                hiddenScreen, ref dest);
        }

        // Scroll Methods
```

```
public static void ResetScroll()
{
    startX = startY = 0;
}

public static void ScrollTo(short newStartX, short newStartY)
{
    startX = newStartX;
    startY = newStartY;
}

public static void ScrollHorizontally(short xDespl)
{
    startX += xDespl;
}

public static void ScrollVertically(short yDespl)
{
    startY += yDespl;
}

// Joystick methods

/** JoystickPressed: returns TRUE if
    *  a certain button in the joystick/gamepad
    *  has been pressed
    */
public static bool JoystickPressed(int boton)
{
    if (!isThereJoystick)
        return false;

    if (Sdl.SDL_JoystickGetButton(joystick, boton) > 0)
        return true;
    else
        return false;
}

/** JoystickMoved: returns TRUE if
    *  the joystick/gamepad has been moved
    *  up to the limit in any direction
    *  Then, int returns the corresponding
    *  X (1=right, -1=left)
    *  and Y (1=down, -1=up)
    */
public static bool JoystickMoved(out int posX, out int posY)
{
    posX = 0; posY = 0;
    if (!isThereJoystick)
        return false;

    posX = Sdl.SDL_JoystickGetAxis(joystick, 0);  // Leo valores (hasta 32768)
    posY = Sdl.SDL_JoystickGetAxis(joystick, 1);
    // Normalizo valores
    if (posX == -32768) posX = -1;  // Normalizo, a -1, +1 o 0
    else if (posX == 32767) posX = 1;
    else posX = 0;
    if (posY == -32768) posY = -1;
    else if (posY == 32767) posY = 1;
    else posY = 0;

    if ((posX != 0) || (posY != 0))
```

```
        return true;
    else
        return false;
}


/** JoystickMovedRight: returns TRUE if
    *  the joystick/gamepad has been moved
    *  completely to the right
    */
public static bool JoystickMovedRight()
{
    if (!isThereJoystick)
        return false;

    int posX = 0, posY = 0;
    if (JoystickMoved(out posX, out posY) && (posX == 1))
        return true;
    else
        return false;
}

/** JoystickMovedLeft: returns TRUE if
    *  the joystick/gamepad has been moved
    *  completely to the left
    */
public static bool JoystickMovedLeft()
{
    if (!isThereJoystick)
        return false;

    int posX = 0, posY = 0;
    if (JoystickMoved(out posX, out posY) && (posX == -1))
        return true;
    else
        return false;
}


/** JoystickMovedUp: returns TRUE if
    *  the joystick/gamepad has been moved
    *  completely upwards
    */
public static bool JoystickMovedUp()
{
    if (!isThereJoystick)
        return false;

    int posX = 0, posY = 0;
    if (JoystickMoved(out posX, out posY) && (posY == -1))
        return true;
    else
        return false;
}


/** JoystickMovedDown: returns TRUE if
    *  the joystick/gamepad has been moved
    *  completely downwards
    */
public static bool JoystickMovedDown()
{
    if (!isThereJoystick)
```

```
            return false;

        int posX = 0, posY = 0;
        if (JoystickMoved(out posX, out posY) && (posY == 1))
            return true;
        else
            return false;
    }


    /** GetMouseX: returns the current X
        *  coordinate of the mouse position
        */
    public static int GetMouseX()
    {
        int posX = 0, posY = 0;
        Sdl.SDL_PumpEvents();
        Sdl.SDL_GetMouseState(out posX, out posY);
        return posX;
    }


    /** GetMouseY: returns the current Y
        *  coordinate of the mouse position
        */
    public static int GetMouseY()
    {
        int posX = 0, posY = 0;
        Sdl.SDL_PumpEvents();
        Sdl.SDL_GetMouseState(out posX, out posY);
        return posY;
    }


    /** MouseClicked: return TRUE if
        *  the (left) mouse button has been clicked
        */
    public static bool MouseClicked()
    {
        int posX = 0, posY = 0;

        Sdl.SDL_PumpEvents();

        // To avoid two consecutive clicks
        int now = Sdl.SDL_GetTicks();
        if (now - lastMouseClick < mouseClickLapse)
            return false;

        // Ahora miro si realmente hay pulsación
        if ((Sdl.SDL_GetMouseState(out posX, out posY) & Sdl.SDL_BUTTON(1)) == 1)
        {
            lastMouseClick = now;
            return true;
        }
        else
            return false;
    }


    // Private (auxiliar) methods

    private static void drawHiddenImage(IntPtr image, int x, int y)
    {
            Sdl.SDL_Rect origin = new Sdl.SDL_Rect(0, 0, width, height);
            Sdl.SDL_Rect dest = new Sdl.SDL_Rect((short)x, (short)y,
                width, height);
            Sdl.SDL_BlitSurface(image, ref origin, hiddenScreen, ref dest);
        }

        private static void drawHiddenImageBackground(IntPtr image, int x, int y)
        {
            Sdl.SDL_Rect origin = new Sdl.SDL_Rect(0, 0, width, 5000);
            Sdl.SDL_Rect dest = new Sdl.SDL_Rect((short)x, (short)y,
                width, height);
            Sdl.SDL_BlitSurface(image, ref origin, hiddenScreen, ref dest);
        }

        /// <summary>
        /// new method for display background
        /// </summary>
        /// <param name="background"></param>
        /// <param name="x"></param>
        /// <param name="y"></param>
        /// <param name="height"></param>
        /// <param name="width"></param>
        public static void DrawHiddenImageBackground(Image background, int x, int y, int
height, int width)
        {

            Sdl.SDL_Rect origin = new Sdl.SDL_Rect(0, 0, (short)width, (short)height);
            Sdl.SDL_Rect dest = new Sdl.SDL_Rect((short)x, (short)y, (short)width,
(short)height);
            Sdl.SDL_BlitSurface(background.GetPointer(), ref origin, hiddenScreen, ref
dest);
        }


        // Alternate key definitions

        public static int KEY_ESC = Sdl.SDLK_ESCAPE;
        public static int KEY_SPC = Sdl.SDLK_SPACE;
        public static int KEY_A = Sdl.SDLK_a;
        public static int KEY_B = Sdl.SDLK_b;
        public static int KEY_C = Sdl.SDLK_c;
        public static int KEY_D = Sdl.SDLK_d;
        public static int KEY_E = Sdl.SDLK_e;
        public static int KEY_F = Sdl.SDLK_f;
        public static int KEY_G = Sdl.SDLK_g;
        public static int KEY_H = Sdl.SDLK_h;
        public static int KEY_I = Sdl.SDLK_i;
        public static int KEY_J = Sdl.SDLK_j;
        public static int KEY_K = Sdl.SDLK_k;
        public static int KEY_L = Sdl.SDLK_l;
        public static int KEY_M = Sdl.SDLK_m;
        public static int KEY_N = Sdl.SDLK_n;
        public static int KEY_O = Sdl.SDLK_o;
        public static int KEY_P = Sdl.SDLK_p;
        public static int KEY_Q = Sdl.SDLK_q;
        public static int KEY_R = Sdl.SDLK_r;
        public static int KEY_S = Sdl.SDLK_s;
        public static int KEY_T = Sdl.SDLK_t;
        public static int KEY_U = Sdl.SDLK_u;
        public static int KEY_V = Sdl.SDLK_v;
        public static int KEY_W = Sdl.SDLK_w;
        public static int KEY_X = Sdl.SDLK_x;
        public static int KEY_Y = Sdl.SDLK_y;
        public static int KEY_Z = Sdl.SDLK_z;
        public static int KEY_1 = Sdl.SDLK_1;
        public static int KEY_2 = Sdl.SDLK_2;
        public static int KEY_3 = Sdl.SDLK_3;
        public static int KEY_4 = Sdl.SDLK_4;
        public static int KEY_5 = Sdl.SDLK_5;
        public static int KEY_6 = Sdl.SDLK_6;
        public static int KEY_7 = Sdl.SDLK_7;
        public static int KEY_8 = Sdl.SDLK_8;
        public static int KEY_9 = Sdl.SDLK_9;
        public static int KEY_0 = Sdl.SDLK_0;
        public static int KEY_UP = Sdl.SDLK_UP;
        public static int KEY_DOWN = Sdl.SDLK_DOWN;
        public static int KEY_RIGHT = Sdl.SDLK_RIGHT;
        public static int KEY_LEFT = Sdl.SDLK_LEFT;
        public static int KEY_RETURN = Sdl.SDLK_RETURN;



    }

}


/**
 * Font.cs - To hide SDL TTF font handling
 *
 * Luis Miguel Rubio Toledo, 2015
 *
 * Changes:
 * 0.01, 06-03-2013: Initial version, based on SdlMuncher 0.14
 */

using System;
using Tao.Sdl;

namespace Tails
{
    class Font
    {
        private IntPtr internalPointer;

        public Font(string fileName, short sizePoints)
        {
            Load(fileName, sizePoints);
        }

        public void Load(string fileName, short sizePoints)
        {
            internalPointer = SdlTtf.TTF_OpenFont(fileName, sizePoints);
            if (internalPointer == IntPtr.Zero)
                Hardware.FatalError("Font not found: " + fileName);
        }

        public IntPtr GetPointer()
        {
            return internalPointer;
        }
    }
}


/**
```

```
 * Image.cs - To hide SDL image handling
 *
 * Luis Miguel Rubio Toledo, 2015
 * Changes:
 * 0.02, 06-03-2013: Initial version
 */
using System;
using Tao.Sdl;


namespace Tails
{
    class Image
    {
        private IntPtr internalPointer;

        public Image(string fileName)  // Constructor
        {
            Load(fileName);
        }

        public void Load(string fileName)
        {
            internalPointer = SdlImage.IMG_Load(fileName);
            if (internalPointer == IntPtr.Zero)
                Hardware.FatalError("Image not found: " + fileName);
        }


        public IntPtr GetPointer()
        {
            return internalPointer;
        }
    }
}


/**
 * Sound.cs
 *
 * Luis Miguel Rubio Toledo, 2015
 *
 * Changes:
 * 0.05  28-05-2015: create class Sound (noly To Do)
 * 0.27  06-06-2015: implement methods
 */

using System;
using Tao.Sdl;

namespace Tails
{
    class Sound
    {
        IntPtr songPointer;

        /// <summary>
        /// constructor
        /// </summary>
        public Sound()
        {

        }
        /// <summary>
        /// constructor with param
        /// </summary>
        /// <param name="fileName"> name file</param>
        public Sound(string fileName)
        {
            songPointer = SdlMixer.Mix_LoadMUS(fileName);
        }

        /// <summary>
        /// use for play once
        /// </summary>
        public void PlayOnce()
        {
            SdlMixer.Mix_PlayMusic(songPointer, 1);
        }

        /// <summary>
        /// use for play Intro
        /// </summary>
        public void PlayIntro()
        {
            SdlMixer.Mix_PlayMusic(songPointer, -1);
        }

        public bool EnableMusic(ref bool onMusic)
        {
            if (onMusic == true)
            {
                onMusic = false;
            }
            else
            {
                onMusic = true;
            }
            return onMusic;
        }
    }
}
/**
 * Sprite.cs - A basic graphic element to inherit from
 *
 * Luis Miguel Rubio Toledo, 2015
 *
 * Changes:
 * 0.03  27-03-2015: create class Sprite
 * 0.05  28-05-2015: draw sprite
 * 0.08  02-05-2015: create methods of colision with other sprite
 * 0.14  04-06-2015: create methods for sequence Sprites
 */

namespace Tails
{
    class Sprite
    {
        // --------------------------------------------
        // Attributes

        protected int x, y;
        protected int startX, startY;
        protected int width, height;
        protected int xSpeed, ySpeed;
        protected Image image;
        protected bool visible;
        protected Image[][] sequence;
        protected bool containsSequence;
        protected int currentFrame;

        protected byte numDirections = 10;
        protected byte currentDirection;
        public const byte RIGHT = 0;
        public const byte LEFT = 1;
        public const byte WAIT = 2;
        public const byte JUMP = 3;
        public const byte CURLRIGTH = 4;
        public const byte CURLLEFT = 5;
        public const byte DIE = 6;


        // --------------------------------------------
        // Methods

        /// <summary>
        /// constructor
        /// </summary>
        public Sprite()
        {

            visible = true;
            sequence = new Image[numDirections][];
            currentDirection = RIGHT;
        }

        /// <summary>
        /// add name image
        /// </summary>
        /// <param name="imageNames">images</param>
        public Sprite(string imageName, int newX, int newY)
        : this()
        {
            LoadImage(imageName);
            x = newX;
            y = newY;
        }



        /// <summary>
        /// Load images
        /// </summary>
        /// <param name="name">Image Name</param>
        public void LoadImage(string name)
        {
            image = new Image(name);
            containsSequence = false;
        }

        /// <summary>
        /// Load Sequence for change image when move
        /// </summary>
        /// <param name="direction">Direction acually</param>
        /// <param name="names">Image names</param>
        public void LoadSequence(byte direction, string[] names)
```

```
{
    int amountOfFrames = names.Length;
    sequence[direction] = new Image[amountOfFrames];
    for (int i = 0; i < amountOfFrames; i++)
        sequence[direction][i] = new Image(names[i]);
    containsSequence = true;
    currentFrame = 0;
}

/// <summary>
/// Load Sequence to change
/// </summary>
/// <param name="names">Image Name</param>
public void LoadSequence(string[] names)
{
    LoadSequence(RIGHT, names);
}

public int GetX()
{
    return x;
}

public int GetY()
{
    return y;
}

public int GetWidth()
{
    return width;
}

public int GetHeight()
{
    return height;
}

public int GetSpeedX()
{
    return xSpeed;
}

public int GetSpeedY()
{
    return ySpeed;
}

public bool IsVisible()
{
    return visible;
}

/// <summary>
/// Check if can move or no
/// </summary>
/// <param name="newX">New position X to draw</param>
/// <param name="newY">New position Y to draw</param>
public void MoveTo(int newX, int newY)
{
    x = newX;
    y = newY;
    if (startX == -1)
    {
        startX = x;
        startY = y;
    }
}

/// <summary>
/// modify Speed x and y
/// </summary>
/// <param name="newXSpeed">new speed X</param>
/// <param name="newYSpeed">new speed Y</param>
public void SetSpeed(int newXSpeed, int newYSpeed)
{
    xSpeed = newXSpeed;
    ySpeed = newYSpeed;
}

/*
public void SetStart(int startX, int startY)
{
    this.startX = startX;
    this.startY = startY;
}
*/

/// <summary>
/// Show elements
/// </summary>
public void Show()
{
    visible = true;
}

/// <summary>
/// Hide elements
/// </summary>
public void Hide()
{
    visible = false;
}

/// <summary>
/// Check colision
/// </summary>
/// <param name="otherSprite">The image to colision</param>
/// <returns>return if have colision or no</returns>
public bool CollisionsWith(Sprite otherSprite)
{
    return (visible && otherSprite.IsVisible() &&
        CollisionsWith(otherSprite.GetX(),
            otherSprite.GetY(),
            otherSprite.GetX() + otherSprite.GetWidth(),
            otherSprite.GetY() + otherSprite.GetHeight()));
}

/// <summary>
/// Check colision
/// </summary>
/// <param name="xStart">Point to start X</param>
/// <param name="yStart">Point to start Y</param>
/// <param name="xEnd">Point to go X</param>
/// <param name="yEnd">Point to go Y</param>
/// <returns>return if have colision</returns>
```

```
public bool CollisionsWith(int xStart, int yStart, int xEnd, int yEnd)
{
    if (visible &&
            (x < xEnd) &&
            (x + width > xStart) &&
            (y < yEnd) &&
            (y + height > yStart)
            )
        return true;
    return false;
}

/// <summary>
/// Draw alls images in the screen
/// </summary>
public void DrawOnHiddenScreen()
{
    if (!visible)
        return;

    if (containsSequence)
        Hardware.DrawHiddenImage(
            sequence[currentDirection][currentFrame], x, y);
    else
        Hardware.DrawHiddenImage(image, x, y);
}

/// <summary>
/// restart x and y
/// </summary>
public void Restart()
{
    x = startX;
    y = startY;
}


/// <summary>
/// move all in background
/// </summary>
public void MoveAllRight()
{

    x += xSpeed;
}

public void MoveAllLeft()
{

    x -= xSpeed;
}
public void MoveAllUp()
{

    y += ySpeed;
}

public void MoveAlldown()
{

    y -= ySpeed;
}
```

```
        /// <summary>
        /// Change frames to draw
        /// </summary>
        public void NextFrame()
        {
            currentFrame++;
            if (currentFrame >= sequence[currentDirection].Length)
                currentFrame = 0;
        }

        /// <summary>
        /// Change direction
        /// </summary>
        /// <param name="newDirection">Neww direction to change</param>
        public void ChangeDirection(byte newDirection)
        {
            if (!containsSequence) return;
            if (currentDirection != newDirection)
            {
                currentDirection = newDirection;
                currentFrame = 0;
            }
        }

        /// <summary>
        ///
        /// </summary>
        /// <returns> current Direction</returns>
        public byte GetCurrentDirection()
        {
            return currentDirection;
        }

        /// <summary>
        /// is virtual void for
        /// </summary>
        public virtual void Animate()
        {

        }

    }
}


/**
 * player.cs
 *
 * Luis Miguel Rubio Toledo, 2015
 *
 * Changes:
 * 0.03  27-03-2015: create class player
 * 0.14  04-06-2015: implement sequence sprites
 * 0.16  05-06-2015: control of time in movements of sprites
 * 0.21  05-06-2015: add lives and method for player
 * 0.22  05-06-2015: create gravity
 * 0.24  05-06-2015: add winLives method
 * 0.26  28-06-2015: create methods animation
 */

using System;
namespace Tails
{
    class Player : Sprite
    {
        protected DateTime dateNow;
        protected int lives;
        bool isJumping = false;
        public Player()
        {
            x = 100;
            y = 290;
            startX = 100;
            startY = 460;
            width = 46;
            height = 32;
            xSpeed = 3;
            ySpeed = 3;
            //LoadImage("data/BlackTails.png");
            lives = 3;

            LoadSequence(RIGHT, new string[] { "tailsSprite/TailsRight_01.png",
            "tailsSprite/TailsRight_02.png", "tailsSprite/TailsRight_03.png",
            "tailsSprite/TailsRight_04.png",  });
            LoadSequence(LEFT, new string[] { "tailsSprite/TailsLeft_01.png",
            "tailsSprite/TailsLeft_02.png", "tailsSprite/TailsLeft_03.png",
            "tailsSprite/TailsLeft_04.png" });
            LoadSequence(WAIT, new string[] { "tailsSprite/TailsWait_01.png",
            "tailsSprite/TailsWait_02.png", "tailsSprite/TailsWait_03.png",
"tailsSprite/TailsWait_04.png",
            "tailsSprite/TailsWait_05.png", "tailsSprite/TailsWait_06.png" });
            LoadSequence(CURLLEFT, new string[] { "tailsSprite/TailsCurlLeft_01.png",
            "tailsSprite/TailsCurlLeft_02.png", "tailsSprite/TailsCurlLeft_03.png",
            "tailsSprite/TailsCurlLeft_04.png"});
            LoadSequence(CURLRIGTH, new string[] { "tailsSprite/TailsCurlsRight_01.png",
            "tailsSprite/TailsCurlsRight_02.png", "tailsSprite/TailsCurlsRight_03.png",
            "tailsSprite/TailsCurlsRight_04.png"});
            LoadSequence(DIE, new string[] { "tailsSprite/TailsDie_01.png",
            "tailsSprite/TailsDie_02.png", "tailsSprite/TailsDie_03.png"});
            LoadSequence(JUMP, new string[] { "tailsSprite/TailsJump_01.png",
            "tailsSprite/TailsJump_02.png", "tailsSprite/TailsJump_03.png",
            "tailsSprite/TailsJump_04.png"});

        }


        /// <summary>
        /// Change direction to Rigth
        /// </summary>
        public void MoveRight()
        {
            ChangeDirection(RIGHT);
            x += xSpeed;
            NextFrame();

        }

        /// <summary>
        /// Change direction to Left
        /// </summary>
        public void MoveLeft()
        {
            ChangeDirection(LEFT);
            x -= xSpeed;
            NextFrame();
        }

        /// <summary>
        /// simule jump To do
        /// </summary>
        public void Jump()
        {
            isJumping = true;
            if (y > 380 - GetHeight())
            {
                ChangeDirection(JUMP);
                y -= ySpeed;
                NextFrame();
            }

        }

        /// <summary>
        /// the player go down
        /// </summary>
        public void fall()
        {
            ChangeDirection(JUMP);
            y += ySpeed;
            NextFrame();
        }

        /// <summary>
        /// simule gravity
        /// </summary>
        public void gravity()
        {

            if (y < 460 && !isJumping)
            {
                fall();
            }
            if (y < 460)
                isJumping = false;

        }

        /// <summary>
        /// active animate
        /// </summary>
        public override void Animate()
        {

            ChangeDirection(WAIT);

            if (DateTime.Now > dateNow.AddMilliseconds(50))
            {
                NextFrame();
                dateNow = DateTime.Now;
            }

        }

        /// <summary>
        /// get live
        /// </summary>
        /// <returns> get Live</returns>
        public int GetLives()
        {
            return lives;
        }


        /// <summary>
        /// When colision with enemies lost one live
        /// </summary>
        /// <returns>count of lives</returns>
        public int LoseLive()
        {
            return lives--;
        }

        /// <summary>
        /// When colision with ring win one live
        /// </summary>
        /// <returns>count of lives</returns>
        public int WinLive()
        {
            return lives++;
        }

        /// <summary>
        /// simule gravity
        /// </summary>
        public void Die()
        {
            ChangeDirection(DIE);
            NextFrame();

        }

        /// <summary>
        /// animation move solo Right
        /// </summary>
        public void AnimateRight()
        {
            MoveRight();
            if (x > 500)
                x = 350;
        }

        /// <summary>
        /// animation move solo Left
        /// </summary>
        public void AnimateLeft()
        {

            MoveLeft();
            if (x < 350)
                x = 500;
        }

        /// <summary>
        /// animation move solo up
        /// </summary>
        public void AnimateJump()
        {
            Jump();
            if (y < 400)
```

```
                y = 500;
            }

        }
    }


    /**
     * Map.cs
     *
     * Luis Miguel Rubio Toledo, 2015
     *
     * Changes:
     * 0.05  28-05-2015: create class Map
     * 0.10  04-06-2015: show map
     * 0.31, 07-06-2015: new parameters
     */

    namespace Tails
    {
        class Map : Sprite
        {
            // protected const int MAXBACKGROUNDS = 5;
            protected Image background;
            // protected Image [] back;
            //protected int x;
            //protected int y;

            public Map()
            {
                //back = new Image[MAXBACKGROUNDS];

                /*for (int i = 0; i < MAXBACKGROUNDS; i++)
                {
                    back[i] = new Image("data/background.png");
                }*/

                background = new Image("data/background.png");
                x = 0;
                y= -150;
                startX = 0;
                startY= -150;
                xSpeed = 2;
                ySpeed = 2;
                height = 1024;
                width = 11264;
            }

            /// <summary>
            /// draw backgraund map
            /// </summary>
            public void Draw()
            {
                /*for (int i = 0; i < MAXBACKGROUNDS; i++)
                {
                    Hardware.DrawHiddenImageBackground(back[i], x, y);
                }*/

                Hardware.DrawHiddenImageBackground(background, x, y, height, width);

            }
        }
    }


    /**
     * Item.cs
     *
     * Luis Miguel Rubio Toledo, 2015
     *
     * Changes:
     * 0.05  28-05-2015: create class Item
     * 0.15  05-06-2015: create movement sprite animation to ring
     * 0.16  05-06-2015: control of time in movements of sprites
     */

    using System;
    namespace Tails
    {

        class Item : Sprite
        {
            DateTime dateNow;
             public Item()
            {
                x = 200;
                y = 480;
                startX = 0;
                startY = 0;
                width = 16;
                height = 16;
                xSpeed = 2;
                ySpeed = 2;
                //LoadImage("data/Ring.png");
                LoadSequence(RIGHT, new string[] { "data/Ring_01.png",
                "data/Ring_02.png", "data/Ring_03.png", "data/Ring_04.png" });

            }

            /// <summary>
            /// use override for animate items
            /// </summary>
             public override void Animate()
             {
                 if (DateTime.Now > dateNow.AddMilliseconds(50))
                 {
                     NextFrame();
                     dateNow = DateTime.Now;
                 }

             }

        }
    }


    /**
     * Enemy.cs
     *
     * Luis Miguel Rubio Toledo, 2015
     *
     * Changes:
     * 0.05  28-05-2015: create class Enemy
     * 0.19  05-06-2015: create sprite animation
     * 0.20  05-06-2015: create method for movement
     */
    using System;
    namespace Tails
    {
        /// <summary>
        /// Enemy class
        /// </summary>
        class Enemy : Sprite
        {
            protected DateTime dateNow;


            public Enemy()
            {

                dateNow = DateTime.Now;
            }

            /// <summary>
            /// use override for animate items
            /// </summary>
            public override void Animate()
            {
                if (DateTime.Now > dateNow.AddMilliseconds(500))
                {
                    NextFrame();
                    dateNow = DateTime.Now;
                }

            }

        }
    }


    /**
     * MotorBug.cs - Partial sonic clone
     *
     * Luis Miguel Rubio Toledo, 2015
     *
     *
     * Changes:
     *
     * 0.27  28-05-2015: create class "Motorbug"
     * 0.27  06-06-2015: inplement methods
     *
     */

    namespace Tails
    {
        class MotorBug : Enemy
        {
            private bool moveLeft = false;

            public MotorBug()
            {
                x = 360;
                y = 460;
                startX = 360;
                startY = 460;
                width = 32;
                height = 26;
                xSpeed = 2;
                ySpeed = 2;
                LoadSequence(RIGHT, new string[] {"data/MotorBugRight_01.png",
                    "data/MotorBugRight_02.png"});
                LoadSequence(LEFT, new string[] {"data/MotorBugLeft_01.png",
                    "data/MotorBugLeft_02.png"});
            }

            /// <summary>
            /// change direction to Right
            /// </summary>
            public void MoveRight()
            {
                ChangeDirection(RIGHT);
                x += xSpeed;
                Animate();
            }


            /// <summary>
            /// Change direction to Left
            /// </summary>
            public void MoveLeft()
            {
                ChangeDirection(LEFT);
                x -= xSpeed;
                Animate();
            }

            /// <summary>
            /// use for move the enemy a section of map
            /// </summary>
            public void MoveAnimation()
            {

                if (!moveLeft)
                {
                    MoveRight();
                    if (x >= 610)
                        moveLeft = true;
                }
                if (moveLeft)
                {
                    MoveLeft();
                    if (x <= 360)
                        moveLeft = false;
                }

            }
        }
    }


    /**
     * CrabMeat.cs - Partial sonic clone
     *
     * Luis Miguel Rubio Toledo, 2015
     *
     *
     * Changes:
```

```
 *
 * 0.27  28-05-2015: create class "Motorbug"
 * 0.27  06-06-2015: inplement methods
 *
 */
namespace Tails
{
    class CrabMeat : Enemy
    {
        public CrabMeat()
        {
            x = 980;
            y = 385;
            startX = 980;
            startY = 385;
            width = 46;
            height = 36;
            xSpeed = 2;
            ySpeed = 2;
            LoadSequence(RIGHT, new string[] {"data/CrabMeat_01.png",
"data/CrabMeat_02.png",
                "data/CrabMeat_03.png", "data/CrabMeat_04.png", "data/CrabMeat_05.png"});
        }
    }
}


/**
 * Animals.cs - Partial sonic clone
 *
 * Luis Miguel Rubio Toledo, 2015
 *
 * Changes:
 * 0.33 07-07-2015: create class Animals and show
 *

 */

using System;
namespace Tails
{
    class Animals : Sprite
    {
        DateTime dateNow;
        private Random rnd;
        private int choose;
        public Animals()
        {
            rnd = new Random();
            choose = rnd.Next(0, 3);

            xSpeed = 4;
            ySpeed = 4;
            LoadSequence(RIGHT, new string[] {"data/Dog_01.png",
                "data/Dog_02.png"});
            LoadSequence(1, new string[] {"data/Pig_01.png",
                "data/Pig_02.png"});
            LoadSequence(2, new string[] {"data/Rabbit_01.png",
                "data/Rabbit_02.png"});
            LoadSequence(3, new string[] {"data/Squirrel_01.png",
                "data/Squirrel_02.png"});
        }
        /// <summary>
        /// animation for animals
        /// </summary>
        public override void Animate()
        {

            switch(choose)
            {
                case 0:
                    ChangeDirection(0);
                    x += xSpeed;
                    TimeFrame();
                    break;
                case 1:
                    ChangeDirection(1);
                    x += xSpeed;
                    TimeFrame();
                    break;
                case 2:
                    ChangeDirection(2);
                    x += xSpeed;
                    TimeFrame();
                    break;
                case 3:
                    ChangeDirection(3);
                    x += xSpeed;
                    TimeFrame();
                    break;
            }

        }

        /// <summary>
        /// time for frame animals
        /// </summary>
        public void TimeFrame()
        {
            if (DateTime.Now > dateNow.AddMilliseconds(500))
            {
                NextFrame();
                dateNow = DateTime.Now;
            }
        }
    }
}


/**
 * Game.cs - Partial sonic clone
 *
 * Luis Miguel Rubio Toledo, 2015
 *
 * Game class: game logic
 *
 * Changes:
 *
 * 0.04, 22-05-2015: create class "Game"
 * 0.05  28-05-2015: show player and enemies
 * 0.06  02-05-2015: move with using the arrow keys left and right
 * 0.07  03-05-2015: show 20 rings consecutives
 * 0.08  02-06-2015: check colision with rings
 * 0.09  02-06-2015: check colision with enemies
 * 0.11  04-06-2015: move rings with map
 * 0.15  05-06-2015: the rings have movement in game
 * 0.20  05-06-2015: the enemies have movement in game
 * 0.21  05-06-2015: Restart game correctly
 * 0.23  06-06-2015: check colision with sprites
 * 0.24  06-06-2015: create struct for CurrentScore and modify
 * 0.28  06-06-2015: implement sound
 * 0.29  06-06-2015: add life when take 10 rings correctly
 * 0.30  06-06-2015: clean and debug in collisions
 * 0.32  07-06-2015: display all screen background
 * 0.33  07-06-2015: show animals when enemy death
 */
using System;


namespace Tails
{
    struct CurrentScore
    {
        public string name;
        public int score;
        public int currenRings;

    }
    class Game
    {
        // --------------------------------------------
        // Attributes
        Sound startGame, extraLive, takeRing;

        const int MAXITEMS = 20;
        const int MAXANIMALS = 2;
        DateTime waitSecons;
        CurrentScore myScore;

        // int score;
        // int life;

        Font font18;
        Map myMap;

        Player player;
        CrabMeat crabMeat;
        MotorBug motorBug;
        Item[] ring;
        Animals[] animal;
        bool finished;

        int startRingX;
        int startRingY;
        bool getLife;


        // --------------------------------------------
        // Methods
        public Game()
        {

            startGame = new Sound("sound/green_hill_zone.mp3");
            extraLive = new Sound("sound/extra_life.mp3");
            takeRing = new Sound("sound/Ring_Sound_Effect.mp3");
            font18 = new Font("data/Joystix.ttf", 18);
            myMap = new Map();
            player = new Player();
            crabMeat = new CrabMeat();
            motorBug = new MotorBug();

            ring = new Item[MAXITEMS];
            startRingX = 150;
            startRingY = 400;

            myScore.currenRings = 0;
            myScore.score = 0;

            //score = 0;
            // life = 3;
            for (int i = 0; i < MAXITEMS; i++)
            {
                ring[i] = new Item();
                ring[i].MoveTo(startRingX, startRingY);

                startRingX += 40;
            }

            getLife = false;

            animal = new Animals[MAXANIMALS];
            animal[0] = new Animals();
            animal[1] = new Animals();
            animal[0].Hide();
            animal[1].Hide();
        }

        /// <summary>
        /// Display alls movements
        /// </summary>
        public void MoveElements()
        {

            for (int i = 0; i < MAXITEMS; i++)
            {
                ring[i].Animate();

            }
            crabMeat.Animate();
            motorBug.MoveAnimation();
            animal[0].Animate();
            animal[1].Animate();

        }

        public void RestartAll()
        {
            startRingX = 150;
            startRingY = 400;
            for (int i = 0; i < MAXITEMS; i++)
            {
                ring[i] = new Item();
                ring[i].MoveTo(startRingX, startRingY);

                startRingX += 40;
            }
            motorBug.Show();
            crabMeat.Show();
            motorBug.Restart();
            crabMeat.Restart();
            myMap.Restart();
            player.Restart();
        }

        /*
        public void StartEnemies()
        {

            crabMeat.MoveTo(200, 480);

            //crabMeat.LoadImage("data/CrabMeat.png");
            motorBug.LoadImage("data/MotorBug.png");

            motorBug.MoveTo(500, 400);

            //crabMeat.DrawOnHiddenScreen();
            motorBug.DrawOnHiddenScreen();

        }
         * */

        /// <summary>
        /// check input
        /// </summary>

        void CheckInput()
        {
            if ((Hardware.KeyPressed(Hardware.KEY_LEFT)))
            {
                if (player.GetX() > 0)
                    player.MoveLeft();
                if (myMap.GetX() < 0 && player.GetX() >= 0)
                {
                    //motorBug.MoveAllRight();
                    crabMeat.MoveAllRight();
                    myMap.MoveAllRight();
                    for (int i = 0; i < MAXITEMS; i++)
                    {
                        ring[i].MoveAllRight();
                    }
                }
                waitSecons = DateTime.Now;
            }

            if ((Hardware.KeyPressed(Hardware.KEY_RIGHT)))
            {
                if (player.GetX() < 600 - player.GetHeight())
                    player.MoveRight();
                //motorBug.MoveAllLeft();
                crabMeat.MoveAllLeft();
                myMap.MoveAllLeft();
                for (int i = 0; i < MAXITEMS; i++)
                {
                    ring[i].MoveAllLeft();
                }
                waitSecons = DateTime.Now;

            }
            if (Hardware.KeyPressed(Hardware.KEY_SPC))
            {
                player.Jump();
                waitSecons = DateTime.Now;
            }

            // ESC to return to Intro
            if (Hardware.KeyPressed(Hardware.KEY_ESC))
                finished = true;

            if (DateTime.Now > waitSecons.AddSeconds(6))
            {
                waitSecons = DateTime.Now;
                player.Animate();
            }
            player.gravity();

        }

        /// <summary>
        /// show all Elements
        /// </summary>
        void DrawElements()
        {
            Hardware.ClearScreen();
            myMap.Draw();


            for (int i = 0; i < MAXITEMS; i++)
            {
                ring[i].DrawOnHiddenScreen();
            }

            motorBug.DrawOnHiddenScreen();
            crabMeat.DrawOnHiddenScreen();
            Hardware.WriteHiddenText("Score: " + myScore.score,
                60, 30,
                0xCC, 0xCC, 0xCC,
                font18);

            Hardware.WriteHiddenText("Rings: " + myScore.currenRings,
                300, 30,
                0xCC, 0xCC, 0xCC,
                font18);

            Hardware.WriteHiddenText("Lives: " + player.GetLives(),
                600, 30,
                0xCC, 0xCC, 0xCC,
                font18);

            animal[0].DrawOnHiddenScreen();
            animal[1].DrawOnHiddenScreen();
            player.DrawOnHiddenScreen();

            Hardware.ShowHiddenScreen();
            Hardware.Pause(10);
        }


        // collisions
        void CheckCollisions(bool onMusic)
```

```
        {

            // collision for enemies
            if (player.CollisionsWith(motorBug) || player.CollisionsWith(crabMeat))
            {
                // direccions sprite 0 = RIGHT, 1 = LEFT, 2 = WAIT
                for (int i = 0; i <= 2; i++)
                {
                    if (myScore.currenRings > 0)
                    {
                        myScore.currenRings = 0;
                        player.MoveTo(player.GetX() - player.GetWidth(), player.GetY());
                        player.Die();
                    }
                    else if (player.GetCurrentDirection() == i && myScore.currenRings == 0)
                    {
                        // life--;
                        player.Die();
                        player.LoseLive();
                        RestartAll();
                        if (player.GetLives() == 0)
                            finished = true;
                    }
                }
                // direccions sprite 3 = JUMP, 4 = CURLRIGTH, 5 = CURLLEFT
                for (int i = 3; i <= 5; i++)
                {
                    if (player.GetCurrentDirection() == i &&
    player.CollisionsWith(motorBug))
                    {
                        myScore.score += 1000;
                        animal[0].MoveTo(motorBug.GetX(), motorBug.GetY());
                        motorBug.Hide();
                        animal[0].Show();

                    }
                    if (player.GetCurrentDirection() == i &&
    player.CollisionsWith(crabMeat))
                    {

                        myScore.score += 1000;
                        animal[1].MoveTo(crabMeat.GetX(), crabMeat.GetY());
                        crabMeat.Hide();
                        animal[1].Show();

                    }

                }
            }


            // when take 10 ring lvl up
            if (myScore.currenRings != 0)
            {

                if (myScore.currenRings % 10 == 0 && !getLife)
                {
                    if (onMusic)
                        extraLive.PlayOnce();
                    getLife = true;
                    player.WinLive();
                }
                if (myScore.currenRings % 2 == 1)
                    getLife = false;
            }

            //collision for items
            for (int i = 0; i < MAXITEMS; i++)
            {
                if (player.CollisionsWith(ring[i]))
                {
                    if (onMusic)
                        takeRing.PlayOnce();
                    myScore.score += 100;
                    myScore.currenRings++;
                    ring[i].Hide();

                }
            }

        }

        /// <summary>
        /// Execute game
        /// </summary>
        public void Run(bool onMusic)
        {
            player.Restart();
            if (onMusic)
                startGame.PlayIntro();
            // Game Loop

            do
            {
                // Update screen
                //StartEnemies();
                MoveElements();
                DrawElements();

                CheckInput();
                CheckCollisions(onMusic);

            }
            while (!finished);

            // Wait for ESC to be released
            while (Hardware.KeyPressed(Hardware.KEY_ESC)) { }
        }
    }
}


/**
 * Intro.cs
 *
 * Luis Miguel Rubio Toledo, 2015
 *
 * Changes:
 * 0.01, 06-03-2015: Initial version
 * 0.04, 22-05-2015: Game can be quit (ESC) or start (SPC)
 * 0.17  05-06-2015: show menu with all cases
 */
using System;
using System.Threading;
```

```
namespace Tails
{
    class Intro
    {
        private Font font18;
        private Font font74;
        private Sprite backgroundTails;
        private Player waiting;
        private Sound intro;

        private bool finished;

        public bool GameChosen { get; set; }
        public bool ExitChosen { get; set; }

        /// <summary>
        /// constructor
        /// </summary>
        public Intro()
        {
            finished = false;
            font18 = new Font("data/Joystix.ttf", 24);
            font74 = new Font("data/Joystix.ttf", 74);
            backgroundTails = new Sprite("data/Tails.png", 450, 100);
            waiting = new Player();
            intro = new Sound("sound/title_screen.mp3");

        }

        public void Run(bool onMusic)
        {
            GameChosen = false;
            ExitChosen = false;
            finished = false;
            if (onMusic)
                intro.PlayIntro();
            do
            {

                ShowIntro();

                if (Hardware.KeyPressed(Hardware.KEY_SPC))
                {

                    // And prepare to enter the game
                    GameChosen = true;
                    finished = true;
                }
                if (Hardware.KeyPressed(Hardware.KEY_S))
                {
                    ScoreScreen scoreScreen = new ScoreScreen();
                    scoreScreen.Run();
                }
                if (Hardware.KeyPressed(Hardware.KEY_C))
                {
                    CreditsScreen creditsScreen = new CreditsScreen();
                    creditsScreen.Run();
                }
                if (Hardware.KeyPressed(Hardware.KEY_H))
                {
                    HelpScreen helpScreen = new HelpScreen();
                    helpScreen.Run();
                }
                if (Hardware.KeyPressed(Hardware.KEY_ESC))
                {
                    ExitChosen = true;
                    finished = true;
                }

            }
            while (!finished);

        }

        /// <summary>
        /// display intro
        /// </summary>
        public void ShowIntro()
        {
            Hardware.ClearScreen();
            waiting.Animate();
            waiting.DrawOnHiddenScreen();
            backgroundTails.DrawOnHiddenScreen();

            Hardware.WriteHiddenText("TAILS! ",
                    70, 100,
                0xCC, 0xCC, 0xCC,
                font74);

            Hardware.WriteHiddenText("Press SPACE to Start! ",
                    50, 400,
                0xCC, 0xCC, 0xCC,
                font18);

            Hardware.WriteHiddenText("Press S to Score! ",
                    50, 450,
                20, 0xCC, 0xCC,
                font18);

            Hardware.WriteHiddenText("Press C to Credits! ",
                    50, 500,
                0xCC, 0xCC, 20,
                font18);

            Hardware.WriteHiddenText("Press H to Help! ",
                    50, 550,
                0xCC, 20, 0xCC,
                font18);
            Hardware.Pause(90); // Not to use a 100% CPU for nothing
            Hardware.ShowHiddenScreen();

        }
    }
}


/**
 * HelpScreen.cs
 *
 * Luis Miguel Rubio Toledo, 2015
 *
 * Changes:
 * 0.05  28-05-2015: create class HelpScreen
 * 0.25  28-06-2015: create menu and method HowPlaying
 * 0.26  28-06-2015: create animation
 */

namespace Tails
{
    /// <summary>
    /// Class HelpScreen
    /// Show how play with animations
    /// </summary>
    class HelpScreen
    {
        private Font font20;
        private Font explain;
        private Image left;
        private Image right;

        Player Left;
        Player Right;
        Player Jump;


        public HelpScreen()
        {
            font20 = new Font("data/Joystix.TTF", 20);
            explain = new Font("data/Joystix.ttf", 15);
            left = new Image("data/arrow_left.png");
            right = new Image("data/arrow_right.png");
            Left = new Player();
            Right = new Player();
            Jump = new Player();

        }


        /// <summary>
        /// Loop to show help to play when press ESC return
        /// </summary>
        public void Run()
        {

            do
            {
                //HEADER
                Hardware.ClearScreen();
                Hardware.WriteHiddenText("HELP - TAILS",
                    350, 100,
                    0x08, 0x6D, 0x08,
                    font20);

                Hardware.WriteHiddenText("Hit option number",
                    330, 150,
                    0x99, 0x99, 0x99,
                    font20);

                //Option 1
                Hardware.WriteHiddenText("1: Move arround map ",
                    320, 300,
                    0, 144, 250,
                    font20);

                //Option 2
                Hardware.WriteHiddenText("2: How Playing? ",
                    350, 350,
                    0, 144, 250,
                    font20);

                Hardware.WriteHiddenText("Hit Q to return",
                    370, 640,
                    0x99, 0x99, 0x99,
                    font20);
                Hardware.ShowHiddenScreen();
                Hardware.Pause(50);

                if (Hardware.KeyPressed(Hardware.KEY_1))
                    Animatios();
                if (Hardware.KeyPressed(Hardware.KEY_2))
                    HowPlaying();
            }
            while (!Hardware.KeyPressed(Hardware.KEY_Q));
        }


        /// <summary>
        /// Move Arround the map
        /// </summary>
        public void Animatios()
        {

            Right.MoveTo(350, 185);
            Jump.MoveTo(650, 500);
            Left.MoveTo(500, 250);

            do
            {
                //HEADER
                Hardware.ClearScreen();
                Hardware.WriteHiddenText("HELP - TAILS - Move arround map",
                    250, 100,
                    0x08, 0x6D, 0x08,
                    font20);

                //Arrow right
                Hardware.WriteHiddenText("Press key ",
                    150, 190,
                    0, 144, 250,
                    font20);
                Hardware.DrawHiddenImage(right, 290, 170);

                Right.DrawOnHiddenScreen();
                Right.AnimateRight();

                //Arrow Left
                Hardware.WriteHiddenText("Press key ",
                    150, 260,
                    0, 144, 250,
                    font20);
                Hardware.DrawHiddenImage(left, 290, 240);

                Left.DrawOnHiddenScreen();
                Left.AnimateLeft();

                //Arrow Jump
                Hardware.WriteHiddenText("When Press key SPACE ",
                    500, 330,
                    0, 144, 250,
                    font20);

                Jump.DrawOnHiddenScreen();
                Jump.AnimateJump();

                Hardware.WriteHiddenText("Hit ESC to return",
                    370, 640,
                    0x99, 0x99, 0x99,
                    font20);
                Hardware.ShowHiddenScreen();
                Hardware.Pause(50);


            } while (!Hardware.KeyPressed(Hardware.KEY_ESC));
        }


        /// <summary>
        /// Explain about playing game
        /// </summary>
        public void HowPlaying()
        {
            do
            {
                //HEADER
                Hardware.ClearScreen();
                Hardware.WriteHiddenText("HELP - TAILS - How Playing?",
                    250, 100,
                    0x08, 0x6D, 0x08,
                    font20);

                // Text of information
                Hardware.WriteHiddenText("The player will use to Tails for the purpose of
complete this level.",
                    130, 220,
                    255, 0, 0,
                    explain);
                Hardware.WriteHiddenText("Tails can move to left, to right, jump, he can do
a curl to kill ",
                    130, 250,
                    255, 0, 0,
                    explain);
                Hardware.WriteHiddenText("the enemies or can to slide. He can run and take
items (rings). ",
                    130, 280,
                    255, 0, 0,
                    explain);
                Hardware.WriteHiddenText("If Tails has any ring and hit him then he lost all
rings.",
                    130, 310,
                    255, 0, 0,
                    explain);
                Hardware.WriteHiddenText("If take 100 ring before hit him then, he will
create a life, ",
                    130, 340,
                    255, 0, 0,
                    explain);
                Hardware.WriteHiddenText("if he has not any ring then he lost a life, and he
will start",
                    130, 370,
                    255, 0, 0,
                    explain);
                Hardware.WriteHiddenText("a new this level. Tails start the game with 3 
lifes.",
                    130, 400,
                    255, 0, 0,
                    explain);
                Hardware.WriteHiddenText("If lives of Tails are 0 then finish the party. ",
                    130, 430,
                    255, 0, 0,
                    explain);

                Hardware.WriteHiddenText("Hit ESC to return",
                    370, 640,
                    0x99, 0x99, 0x99,
                    font20);
                Hardware.ShowHiddenScreen();
                Hardware.Pause(50);

            } while (!Hardware.KeyPressed(Hardware.KEY_ESC));
        }
    }

}


/**
 * ScoreScreen.cs
 *
 * Luis Miguel Rubio Toledo, 2015
 *
 * Changes:
 * 0.05  28-05-2015: create class ScoreScreen
 * 0.13  04-06-2015: show ScoreScreen
 * 0.31  06-06-2015: implement arraList and load file in class ScoreScreen
 */

using System;
using System.IO;
using System.Collections;

/// <summary>
/// Class ScoreScreen
/// Show Score
/// </summary>
namespace Tails
{
    class ScoreScreen
    {

        Font font18;
        StreamReader Reader;
        ArrayList scores;
        string line;

        public ScoreScreen()
        {

            font18 = new Font("data/Joystix.ttf", 18);
            scores = new ArrayList();
        }
        /// <summary>
        /// Draw Scrore in the screen
        /// Draw lives
        /// </summary>
        public void Run()
        {
            try
            {

                Reader = File.OpenText("data/Scores.txt");
                do
                {
                    line = Reader.ReadLine();
                    if (line != null)
                        scores.Add(line);
                } while (line != null);
                Reader.Close();
            }
            catch (Exception)
            {
            }
            scores.Sort();
            scores.Reverse();

            do
            {
                Hardware.ClearScreen();
                Hardware.WriteHiddenText("Score - TAILS",
                    350, 100,
                    0x08, 0x6D, 0x08,
                    font18);

                //Names
                short posY = 190;
                foreach (string punt in scores)
                {
                    Hardware.WriteHiddenText((string)punt, 370, posY, 0xCC, 0xCC, 0xCC, 
font18);
                    posY += 40;
                }

                Hardware.WriteHiddenText("Hit Q to return",
                    370, 640,
                    0x99, 0x99, 0x99,
                    font18);
                Hardware.ShowHiddenScreen();
                Hardware.Pause(50);
            }
            while (!Hardware.KeyPressed(Hardware.KEY_Q));
        }

    }
}


/**
 * CreditsScreen.cs
 *
 * Luis Miguel Rubio Toledo, 2015
 *
 * Changes:
```

```
 * 0.05  28-05-2015: create class CreditScreen
 * 0.12  04-06-2015: show CreditScreen
 * 0.18  05-06-2015: create swich with 4 effects
 */
using System;
namespace Tails
{
    /// <summary>
    /// Class CreditsScreen
    /// Show screen of credits
    /// </summary>
    class CreditsScreen
    {
        private Font font18;
        Random rnd;
        private string programmer;
        short y;
        float redColour;      //float to
        float greenColour;    //increment
        float blueColour;     //colour by 0,5
        byte redColourEsc;
        byte greenColourEsc;

        public CreditsScreen()
        {
            rnd = new Random();
            font18 = new Font("data/Joystix.TTF", 25);
            programmer = "Luis Miguel Rubio Toledo";
            redColourEsc  = 255;
            greenColourEsc = 0;
            redColour = 0;
            greenColour = 0;
            blueColour = 0;
            y = 550;
        }


        /// <summary>
        /// Loop that show the name to programers
        /// </summary>
        public void Run()
        {

            int effect;
            effect = rnd.Next(0, 4);
            switch (effect)
            {
                case 0: Effect0(); break;
                case 1: Effect1(); break;
                case 2: Effect2(); break;
                case 3: Effect3(); break;
                case 4: Effect4(); break;
                //case 5: Effect5(); break;
            }



        }

        /// <summary>
        /// first version
        /// </summary>
        public void Effect0()
        {
            do
            {

                //HEADER
                Hardware.ClearScreen();
                Hardware.WriteHiddenText("Credits - TAILS",
                    350, 100,
                    0x08, 0x6D, 0x08,
                    font18);

                //Name
                Hardware.WriteHiddenText(programmer,
                    150, 190,
                    0, 0, 250,
                    font18);

                //Draw "Hit Q Return" changing colour
                Hardware.WriteHiddenText("Hit Q to return",
                    Convert.ToInt16(512 - "Hit Q to return".Length * 14 / 2), 600, //Center
text(x,y)
                    redColourEsc, greenColourEsc, 0,
                    font18);
                Hardware.ShowHiddenScreen();
                Hardware.Pause(10);
                greenColourEsc += 5;
                redColourEsc -= 5;
            }
            while (!Hardware.KeyPressed(Hardware.KEY_Q));
        }

        //Effect1: moving up
        public void Effect1()
        {

            while ((y > 100) && (!Hardware.KeyPressed(Hardware.KEY_Q)))
            {

                Hardware.ClearScreen();
                //Draw moving up programmers
                Hardware.WriteHiddenText(programmer,
                    Convert.ToInt16(512 - programmer.Length * 14 / 2), y, //Center text(x,y)
                    255, 255, 255,
                    font18);

                //Draw "Hit Q Return" changing colour
                Hardware.WriteHiddenText("Hit Q to return",
                    Convert.ToInt16(512 - "Hit Q to return".Length * 14 / 2), 600, //Center
text(x,y)
                    redColourEsc, greenColourEsc, 0,
                    font18);
                Hardware.ShowHiddenScreen();
                Hardware.Pause(10);
                greenColourEsc += 5;
                redColourEsc -= 5;
                y--; ;
            }

        }

        //Effect2: moving up + black&white
        public void Effect2()
        {
            while ((y > 100) && (!Hardware.KeyPressed(Hardware.KEY_Q)))
            {
                Hardware.ClearScreen();

                //Draw moving up programmers black&white
                Hardware.WriteHiddenText(programmer,
                    Convert.ToInt16(512 - programmer.Length * 14 / 2), y, //Center text(x,y)
                    (byte)redColour, (byte)greenColour, (byte)blueColour,
                    font18);
                redColour += 0.5F;
                greenColour += 0.5F;
                blueColour += 0.5F;

                //Draw "Hit Q Return" changing colour
                Hardware.WriteHiddenText("Hit Q to return",
                    Convert.ToInt16(512 - "Hit Q to return".Length * 14 / 2), 600, //Center
text(x,y)
                    redColourEsc, greenColourEsc, 0,
                    font18);
                Hardware.ShowHiddenScreen();
                Hardware.Pause(10);
                greenColourEsc += 5;
                redColourEsc -= 5;
                y--;
            }

        }

        //Effect3: moving down
        public void Effect3()
        {
            y = 100;

            do
            {

                Hardware.ClearScreen();
                //Draw moving up programmers blur
                Hardware.WriteHiddenText(programmer,
                    Convert.ToInt16(512 - programmer.Length * 20 / 2), y, //Center text(x,y)
                    255, 255, 255,
                    font18);

                //Draw "Hit Q Return" changing colour
                Hardware.WriteHiddenText("Hit Q to return",
                    Convert.ToInt16(512 - "Hit Q to return".Length * 14 / 2), 600, //Center
text(x,y)
                    redColourEsc, greenColourEsc, 0,
                    font18);
                Hardware.ShowHiddenScreen();
                Hardware.Pause(10);
                greenColourEsc += 5;
                redColourEsc -= 5;
                if(y < 550)
                    y++;
            } while (!Hardware.KeyPressed(Hardware.KEY_Q));

        }


        //Effect4: moving up + blue
        public void Effect4()
        {

            while ((y > 100) && (!Hardware.KeyPressed(Hardware.KEY_Q)))
            {
                Hardware.ClearScreen();
                //Draw moving up programmers blur black&white
                Hardware.WriteHiddenText(programmer,
                    Convert.ToInt16(512 - programmer.Length * 20 / 2), y, //Center text(x,y)
                    (byte)redColour, (byte)greenColour, (byte)blueColour,
                    font18);

                blueColour += 0.5F;

                //Draw "Hit Q Return" changing colour
                Hardware.WriteHiddenText("Hit Q to return",
                    Convert.ToInt16(512 - "Hit Q to return".Length * 14 / 2), 600, //Center
text(x,y)
                    redColourEsc, greenColourEsc, 0,
                    font18);
                Hardware.ShowHiddenScreen();
                Hardware.Pause(10);
                greenColourEsc += 5;
                redColourEsc -= 5;
                y--;
            }

        }
    }
}
```