# Contents

## 1 World Dimension

1.1 What Is a World? 3  
1.2 Why Do We Use the World? 3  

## 2 Viewport

## 3 Bitmap

3.1 What is a Bitmap? 3  
3.2 RGB Intensity 3  

## 4 Background

4.1 What Is a Background? 3  
4.2 Position 3  

## 5 Animations

5.1 What Is an Animation? 4  
5.2 What Do We Do When the Animation Ends? 4  

## 6 Frames

6.1 What Is a Frame? 4  
6.2 Transparency 4  

## 7 Sprite

7.1 What Is a Sprite? 4  
7.2 Active 5  
7.3 Visible 5  
7.4 Animation 5  
8 ZOrder 5  

## Game Implementation

Step 1: Adding the Background 5  
Step 2: Adding Sprites 6  
Step 3: Adding the “Trooper” Sprite 8  
Step 4: Adding the “Condor” Sprite 9  
Step 5: Adding Another Animation “Explosion” to the “Condor” Sprite 10
1 WORLD DIMENSION

1.1 WHAT IS A WORLD?

The world is the environment where the entire game will take place. It is a rectangle with a defined width and height.

1.2 WHY DO WE USE THE WORLD?

The simulation application applies mathematical transformations on game objects and cameras. These transformations operate on game object structures. All objects must be within a defined area. If a position of an object is outside the world, we can reposition the object or choose another option.

2 VIEWPORT

A world rectangular coordinate selected for display is called a window. (Note: This is not the Operating System Window.) The window defines what is viewed, whereas the viewport defines where it is to be displayed. A viewport is a rectangular area of the display window. By default, it is the entire window (client area of the operating system’s window) application. It can be set to any smaller size in pixels.

3 BITMAP

3.1 WHAT IS A BITMAP?

A bitmap is an image made up of thousands of pixels. Every pixel has its own position and qualities. Qualities such as color and brightness are stored in the bitmap file.

3.2 RGB INTENSITY

Monitors work with red, green, and blue signals. Every colored pixel contains a value of red, green, and blue within it. The intensity of every basic color creates a color. In other words, colors are displayed as varying intensities of red, green, and blue dots.

4 BACKGROUND

4.1 WHAT IS A BACKGROUND?

A background is an image or bitmap that will be placed in the world at a certain position. The area of the screen display that is not covered by a game object is the background that can be seen by the player. Game objects are displayed on top of a background.

4.2 POSITION

The background should be positioned within the world. The background’s position is situated relatively to the world rectangle, not relatively to the viewport.
5 Animations

5.1 What Is an Animation?

An animation is a series of images drawn in sequence with a specific delay for each image. Each frame of an animation is slightly different. When played in succession, they give the appearance of movement.

5.2 What Do We Do When the Animation Ends?

When the animation ends, it can:

- Restart from the first frame.
- Restart from a certain frame (not necessarily the first one).
- Remain at the last frame.

6 Frames

6.1 What Is a Frame?

A frame is simply an image that is part of an animation.

6.2 Transparency

A frame is made out of a bitmap, which is a matrix of points that has a rectangular or square shape. Of course, not all images have a rectangular shape. Areas not used in the rectangle will be filled with one specific color called the transparency color. The drawing mechanism will ignore all pixels having the transparency color; in other words, the transparent pixels will not be drawn.

7 Sprite

7.1 What Is a Sprite?

Sprites are the pictures displayed on top of the background. They are also called game objects. Usually, sprites represent all the moving parts of the game. Sprites can be:

- Enemies
- Animated parts of the background
- Bullets
- Any other moving game object

Since the sprite status varies in terms of images, position, and values, the sprite needs to save information such as:

- Current position
- Current speed
- Current direction
- Visibility status
- Current frame
- Current animation
7.2 Active

This specifies whether the sprite will be active. When the sprite is not active, all its behaviors are disabled and it is not visible. Then the sprite will not be processed during the game loop. An inactive sprite is usually used when a sprite prototype is needed. A sprite prototype can be used to create identical sprites at run time. Inactive sprites are also used when a sprite does not need to be present when the level starts.

7.3 Visible

The code will process all the game objects with every game loop. Processing game objects includes handling and drawing. When the sprite is visible, it will be handled and drawn. However, when the sprite is invisible, it will be handled but not drawn.

7.4 Animation

The sprite or the game object requires an animation to be represented visually. During the process of the sprite creation, an animation with one or more frames should be specified.

8 ZOrder

To understand ZOrder, we have to simulate a 3D coordinates axis. Look at the diagram below and take note of the x-, y-, and z-axes. The z-axis specifies the depth axis, and the greater the value of the ZOrder, the “deeper” the object is to the screen. For example, the picture with the character has a lower ZOrder (1) while the picture with the ball has a higher ZOrder (2). Therefore, the ball appears to be behind the character. Because it is a 2D application, the ZOrder only affects the displacement order of pictures, without affecting their size. All the game objects are affected by the ZOrder.

GAME IMPLEMENTATION

Step 1: Adding the Background

The game concept (theory) is the same for all games, but sometimes there are special cases. In Star Trooper, the special case is that the background object is two instances of class Background, which is derived from Sprite class. In other words, two sprites will play the role of a scrolling background. In order to add a background, we have to add a sprite.
**Step 2: Adding Sprites**

The sprite’s visual representation in the game is the animation, which is made of one or more frames. Each frame is made of one picture (image) and a delay. An image is a physical bitmap.

- Each picture is a class **Picture** object that holds the bitmap.

  ```csharp
  Picture background = new Picture("Background.bmp", 
  Color.FromArgb(0,255,0));
  ``

  Where: The first argument is the name of the bitmap.  
  The second argument is the transparency color.

- The **Picture** object is added to the game in order to be used by the frame.

  ```csharp
  Game.Add(background);
  ``

- The frame is a class **Frame** object that uses the picture object.

  ```csharp
  Frame backGroundFrame = new Frame(background, 0);
  ``

  Where: The first argument is a picture object.  
  The second argument is the frame delay.

- The animation is a class **Animation** object that is composed of many frames.

  ```csharp
  Animation backGroundAnimation = new Animation();
  ``

- Each **Frame** is added to the animation object.

  ```csharp
  backGroundAnimation.Add(backGroundFrame);
  ``

- Create the first background object “bg” that is derived from the **Sprite** class.

  ```csharp
  Background bg = new Background();
  ``

- Add the animation object already created.

  ```csharp
  bg.Add(backGroundAnimation);
  ``

- Set the sprite position.

  ```csharp
  bg.Position = new Point(320, 240);
  ``

- Add the sprite to the game.

  ```csharp
  Game.Add(bg);
  ``

- Set the sprite Size to fit the screen.

  ```csharp
  bg.ScaleX = 640.0f / background.Width;
  bg.ScaleY = 480.0f / background.Height;
  ``

- Set the background order to 10.

  ```csharp
  bg.ZOrder = 10;
  ```
- Add the sprite to the game.
  
  ```csharp
  Game.Add(bg);
  ```

- Create the second background object “bg2” that is derived from the Sprite class.
  
  ```csharp
  Background bg2 = new Background();
  ```

- Add the animation object already created.
  
  ```csharp
  bg2.Add(backGroundAnimation);
  ```

- Set the sprite position.
  
  ```csharp
  bg2.Position = new Point(320, 240);
  ```

- Set the sprite Size to fit the screen.
  
  ```csharp
  bg2.ScaleX = 640.0f / background.Width;
  bg2.ScaleY = 480.0f / background.Height;
  ```

- Set the background order to 10.
  
  ```csharp
  bg2.ZOrder = 10;
  ```

- Add the sprite to the game.
  
  ```csharp
  Game.Add(bg2);
  ```

- **Note:** The path of the directory holding the pictures as BMP files, should be specified in the constructor of the StarTrooperGame class.

  ```csharp
  PicturesPath = Application.StartupPath+"\StarTrooperResources\Pictures\";
  ```

- Type the following code in the ‘StarTrooperGame.cs’ file in order to add the background to the game.

  ```csharp
  Picture background = new Picture("Background.bmp", Color.FromArgb(0, 255, 0));
  Game.Add(background);
  
  Frame backGroundFrame = new Frame(background, 0);
  Animation backGroundAnimation = new Animation();
  backGroundAnimation.Add(backGroundFrame);
  
  Background bg = new Background();
  bg.Add(backGroundAnimation);
  bg.Position = new Point(320, 240);
  bg.ScaleX = 640.0f / background.Width;
  bg.ScaleY = 480.0f / background.Height;
  bg.ZOrder = 10;
  Game.Add(bg);
  
  Background bg2 = new Background();
  bg2.Add(backGroundAnimation);
  bg2.Position = new Point(320, -240);
  bg2.ScaleX = 640.0f / background.Width;
  bg2.ScaleY = 480.0f / background.Height;
  bg2.ZOrder = 10;
  Game.Add(bg2);
  ```
### Step 3: Adding the “Trooper” Sprite

- Type the following code in the ‘StarTrooperGame.cs’ file in order to create a sprite prototype.

```csharp
public static Trooper Trooper;
```

- Each picture is a class Picture object that holds the bitmap.

```csharp
Picture trooper01 = new Picture("trooper01.bmp", Color.FromArgb(0, 255, 0));
```

Where: The first argument is the name of the bitmap. The second argument is the transparency color.

- The Picture object is added to the game in order to be used by the frame.

```csharp
Game.Add(trooper01);
```

- The frame is a class Frame object that uses the picture object.

```csharp
Frame afTrooper01 = new Frame(trooper01, 5); 
```

Where: The first argument is a picture object. The second argument is the frame delay.

- The animation is a class Animation object that is composed of many frames.

```csharp
Animation trooperAnimation = new Animation();
```

- Each Frame is added to the animation object.

```csharp
trooperAnimation.Add(afTrooper01);
```

- Start playing the animation object.

```csharp
trooperAnimation.Play();
```

- Set the loop property of the animation object to be true in order to loop continuously.

```csharp
trooperAnimation.Loop = true;
```

- Create a trooper sprite object.

```csharp
Trooper trooper = new Trooper();
```

- Add the animation object already created.

```csharp
trooper.Add(trooperAnimation);
```

- Set the sprite position.

```csharp
trooper.Position = new Point(320, 450);
```

- Add the sprite to the game.
Game.Add(trooper);
Trooper = trooper;

- **Note:** The path of the directory holding the pictures as BMP files, should be specified in the constructor of the StarTrooperGame class.

PicturesPath = Application.StartupPath+"\StarTrooperResources\Pictures\";

- Type the following code in the 'StarTrooperGame.cs' file in order to add two sprites to the game.

```csharp
Picture trooper01 = new Picture("trooper01.bmp", Color.FromArgb(0, 255, 0));
Game.Add(trooper01);
Picture trooper02 = new Picture("trooper02.bmp", Color.FromArgb(0, 255, 0));
Game.Add(trooper02);
Picture trooper03 = new Picture("trooper03.bmp", Color.FromArgb(0, 255, 0));
Game.Add(trooper03);
Picture trooper04 = new Picture("trooper04.bmp", Color.FromArgb(0, 255, 0));
Game.Add(trooper04);
Picture trooper05 = new Picture("trooper05.bmp", Color.FromArgb(0, 255, 0));
Game.Add(trooper05);
Picture trooper06 = new Picture("trooper06.bmp", Color.FromArgb(0, 255, 0));
Game.Add(trooper06);

Frame afTrooper01 = new Frame(trooper01, 5);
Frame afTrooper02 = new Frame(trooper02, 5);
Frame afTrooper03 = new Frame(trooper03, 5);
Frame afTrooper04 = new Frame(trooper04, 5);
Frame afTrooper05 = new Frame(trooper05, 5);
Frame afTrooper06 = new Frame(trooper06, 5);

Animation trooperAnimation = new Animation();
trooperAnimation.Add(afTrooper01);
trooperAnimation.Add(afTrooper02);
trooperAnimation.Add(afTrooper03);
trooperAnimation.Add(afTrooper04);
trooperAnimation.Add(afTrooper05);
trooperAnimation.Add(afTrooper06);
trooperAnimation.Play();
trooperAnimation.Loop = true;

Trooper trooper = new Trooper();
trooper.Add(trooperAnimation);
trooper.Position = new Point(320, 450);
Game.Add(trooper);
```

### Step 4: Adding the “Condor” Sprite

- Type the following code in the 'StarTrooperGame.cs' file in order to create a sprite prototype.

```csharp
public static Condor Condor;
```

- Each picture is a class Picture object that holds the bitmap.

```csharp
Picture condor01 = new Picture("condor01.bmp", Color.FromArgb(0, 255, 0));
```

Where: The first argument is the name of the bitmap. The second argument is the transparency color.

- The Picture object is added to the game in order to be used by the frame.

```csharp
Game.Add(condor01);
```

- The frame is a class Frame object that uses the picture object.
Frame afcondor01 = new Frame(condor01, 5);

Where: The first argument is a picture object. The second argument is the frame delay.

- The animation is a class Animation object that is composed of many frames.
  
  Animation condorAnimation = new Animation();

- Each Frame is added to the animation object.
  
  condorAnimation.Add(afcondor01);

- Start playing the animation object.
  
  condorAnimation.Play();

- Set the loop property of the animation object to be true in order to loop continuously.
  
  condorAnimation.Loop = true;

**STEP 5: Adding Another Animation “Explosion” to the “Condor” Sprite**

- Each picture is a class Picture object that holds the bitmap.
  
  Picture condorExplosion01 = new Picture("condorExplosion01.bmp", Color.FromArgb(0, 255, 0));

Where: The first argument is the name of the bitmap. The second argument is the transparency color.

- The Picture object is added to the game in order to be used by the frame.
  
  Game.Add(condorExplosion01);

- The frame is a class Frame object that uses the picture object.
  
  Frame afcondorExplosion01 = new Frame(condorExplosion01, 4);

Where: The first argument is a picture object. The second argument is the frame delay.

- The animation is a class Animation object that is composed of many frames.
  
  Animation condorExplosion = new Animation();

- Each Frame is added to the animation object.
  
  condorExplosion.Add(afcondorExplosion01);

- Start playing the animation object.
  
  condorExplosion.Play();

- Create a condor sprite object.
  
  Condor condor = new Condor();
- Add the animation to the object already created.
  
  ```csharp
  condor.Add(condorAnimation);
  ```

- Add the explosion animation to condor sprite.
  
  ```csharp
  condor.Add(condorExplosion);
  ```

- Add the sprite to the game.
  
  ```csharp
  Condor = condor;
  ```

- Type the following code in the 'StarTrooperGame.cs' file in order to add two sprites to the game.

```csharp
Picture condor01 = new Picture("condor01.bmp", Color.FromArgb(0, 255, 0));
Game.Add(condor01);
Picture condor02 = new Picture("condor02.bmp", Color.FromArgb(0, 255, 0));
Game.Add(condor02);
Picture condor03 = new Picture("condor03.bmp", Color.FromArgb(0, 255, 0));
Game.Add(condor03);
Picture condor04 = new Picture("condor04.bmp", Color.FromArgb(0, 255, 0));
Game.Add(condor04);
Frame afcondor01 = new Frame(condor01, 5);
Frame afcondor02 = new Frame(condor02, 5);
Frame afcondor03 = new Frame(condor03, 5);
Frame afcondor04 = new Frame(condor04, 5);
Animation condorAnimation = new Animation();
condorAnimation.Add(afcondor01);
condorAnimation.Add(afcondor02);
condorAnimation.Add(afcondor03);
condorAnimation.Add(afcondor04);
condorAnimation.Play();
condorAnimation.Loop = true;

Picture condorExplosion01 = new Picture("condorExplosion01.bmp", Color.FromArgb(0, 255, 0));
Game.Add(condorExplosion01);
Picture condorExplosion02 = new Picture("condorExplosion02.bmp", Color.FromArgb(0, 255, 0));
Game.Add(condorExplosion02);
Picture condorExplosion03 = new Picture("condorExplosion03.bmp", Color.FromArgb(0, 255, 0));
Game.Add(condorExplosion03);
Frame afcondorExplosion01 = new Frame(condorExplosion01, 4);
Frame afcondorExplosion02 = new Frame(condorExplosion02, 3);
Frame afcondorExplosion03 = new Frame(condorExplosion03, 4);
Animation condorExplosion = new Animation();
condorExplosion.Add(afcondorExplosion01);
condorExplosion.Add(afcondorExplosion02);
condorExplosion.Add(afcondorExplosion03);
condorExplosion.Play();

Condor condor = new Condor();
condor.Add(condorAnimation);
condor.Add(condorExplosion);
Condor = condor;
```
Note: In this game, by default, sprite classes are created for you in order to only create an object from them (Trooper, Condor, and Fire (later)). Therefore, in order to add a new sprite class (for example, a car) add a file called ‘car.cs’ and create a class called “car” that is derived from the Sprite class.

```csharp
#region Using directives
using System;
using System.Collections.Generic;
using System.Drawing;
using Microsoft.DirectX.DirectInput;
using Microsoft.DirectX;
#endregion
using System.Windows.Forms;

namespace StarTrooper
{
    public class Car: Sprite
    {
        public Car()
        {
        }

        protected Car(Car car): base(car)
        {
        }

        public override Object Clone()
        {
            return new Car(this);
        }

        public override void Update()
        {
        }
    }
}
```

Note: Using our game engine, you can even create a new game (totally different storyline) by adding a file called ‘yourgame.cs’ in which you create a class called “yourgame” derived from the Game class.
Under the ‘program.cs’ you should create an instance of “YourGame.”

```csharp
#region Using directives
using System;
using System.Windows.Forms;
#endregion
namespace StarTrooper
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            using (YourGame yourGame = new YourGame())
                yourGame.Run();
        }
    }
}
```