# C# .NET

이종욱

# 목차

- 기획
- 설계
- 구현
- 실행

# 기획

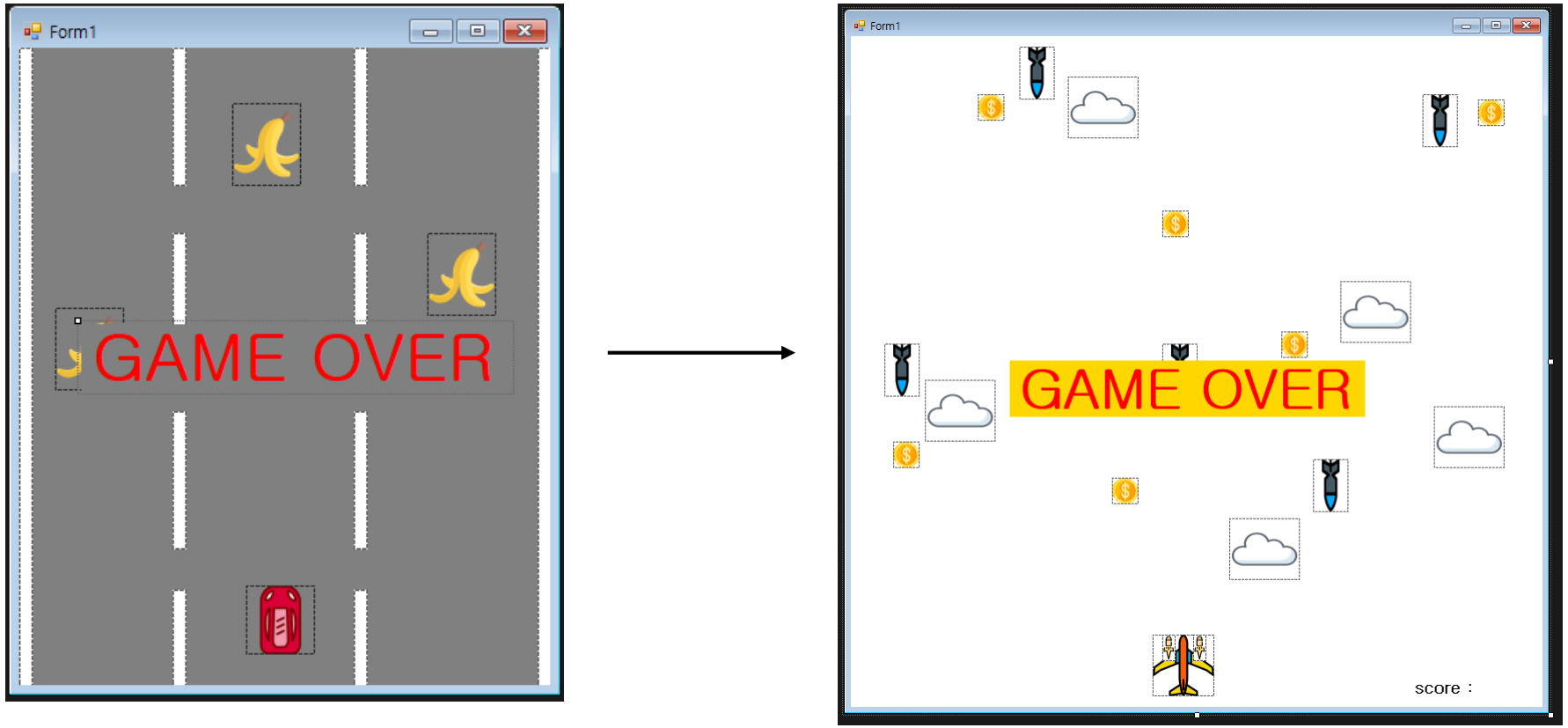
- 평소에 게임을 만들고 싶었다.
- 구글링을 하다가 레이싱 게임 코드 발견했다.
- 블로그 보고 1회 따라서 해보았는데 생각보다 어렵지 않게 느껴져서 응용을 해야겠다고 생각했다.

# 설계

- 만들어진 코드를 기반으로 분석하면서 넣어보고 싶었던 기능들을 추가하였다.
- 추가 하고 싶은 기능들을 메모장에 가상의 조건문을 만들었다.
- 이미지 파일들은 구글에 무료 이미지 사이트에서 구했다.

# 설계 – 이미지, 게임 크기 변경

- 이미지들을 변경하고 가로, 세로 값을 더 넓게 변경하였다.

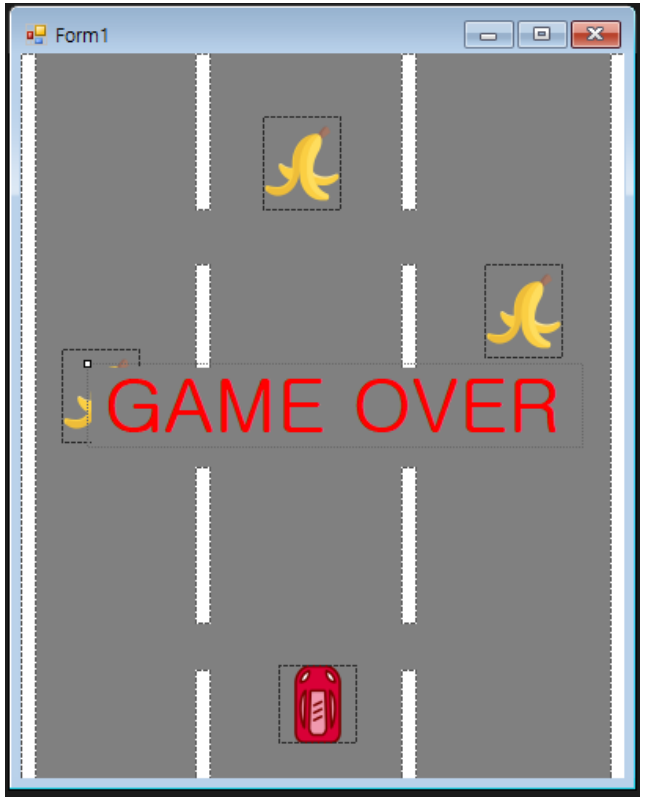


# 설계 – 키보드 이동 범위

- 키보드의 이동 범위를 수정했다.

  원래 레이싱 게임은 키 다운을 유지하고 있으면 움직이는 방식으로 키 딜레이가 생겨서 조작이 어려웠다.

  고정된 값으로 이동을 하여 연타로 조작하는게 쉽고 재미 있을 것 같아 수정하였다.



# 설계 – 키보드 이동 범위

- 왼쪽 키를 누르면 왼쪽으로 비행기 이미지를 왼쪽에서 -50만큼 이동한다.
- 오른쪽을 누르면 비행기 이미지를 왼쪽에서 +50만큼 이동한다.

  오른쪽 이동할 때는 비행기가 화면을 넘어가서 최댓값을 비행기 크기의 반 정도를 남기게 설정하였다.
- 위쪽키는 왼쪽키와, 아래키는 오른쪽과 동일하게 설정하였다.

```
참조 1개
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Left)
    {
        if (airplaneImg.Left > 20)
        {
            airplaneImg.Left += -50;
        }
    }
    else if (e.KeyCode == Keys.Right)
    {
        if (airplaneImg.Right < 800 - airplaneImg.Width / 2)
        {
            airplaneImg.Left += 50;
        }
    }
    else if (e.KeyCode == Keys.Up)
    {
        if (airplaneImg.Top > 20)
        {
            airplaneImg.Top += -50;
        }
    }
    else if (e.KeyCode == Keys.Down)
    {
        if (airplaneImg.Bottom <= 730)
        {
            airplaneImg.Top += 50;
        }
    }
}
```

# 설계 – 미사일 수정

- 미사일의 범위, 개수, 높이도 수정했다.

  위에서 내려오는 미사일이 form1에 사진을 놓은 곳을 기준으로 x값만 이동하고  높낮이는 고정 된 값으로 떨어졌다.

  실제 게임과 같이 다채로운 장애물들이 나왔으면 좋을 것 같아서 다르게 설정하였다.

# 설계 – 미사일 수정

- 미사일 각각 x값을 지정해줘서 지정된 x값 안에서 랜덤으로 나오게 하였다.
- Y값으로는 0부터 10까지 설정해서 미사일이 고정된 y값이 아닌 랜덤으로 나오게 설정하였다.
- 내가 쏜 미사일과 충돌하면 리셋이 되어 다시 랜덤으로 미사일이 내려오게 하였다.
- 랜덤 함수와 x, y값은 미사일 뿐만 아니라 코인, 뒷 배경(구름)에도 사용되기 때문에 전역 함수, 전역 변수로 설정하였다.

```
Random random = new Random();
int x;
int y;
```

```
void Missile(int speed)
{
    if (missile1.Top >= 800)
    {
        x = random.Next(0, 160);
        y = random.Next(0, 10);
        missile1.Location = new Point(x, y);
    }
    else if (missile1.Bounds.IntersectsWith(myMissile.Bounds))
    {
        x = random.Next(0, 160);
        y = random.Next(0, 10);
        missile1.Location = new Point(x, y);
    }
    else if (missile1.Bounds.IntersectsWith(myMissile2.Bounds))
    {
        x = random.Next(0, 160);
        y = random.Next(0, 10);
        missile1.Location = new Point(x, y);
    }
    else
    {
        missile1.Top += speed;
    }

    if (missile2.Top >= 800)
    {
        x = random.Next(160, 320);
        y = random.Next(0, 30);
        missile2.Location = new Point(x, y);
    }
    else if (missile2.Bounds.IntersectsWith(myMissile.Bounds))
    {
        x = random.Next(160, 320);
        y = random.Next(0, 10);
        missile2.Location = new Point(x, y);
    }
    else if (missile2.Bounds.IntersectsWith(myMissile2.Bounds))
    {
        x = random.Next(160, 320);
        y = random.Next(0, 10);
        missile2.Location = new Point(x, y);
    }
    else
    {
        missile2.Top += speed;
    }
```

# 설계 – 코인 시스템

- 코인 시스템과 점수 시스템을 추가하였다.

  미사일과 플레이어의 비행기가 부딪치면 게임이 종료되는 코드를 반대로 구현해 코인과 비행기가 충돌하면 점수 label에 숫자가 표시되게 기능을 추가하였다.

  미사일과 마찬가지로 x값과 y값을 랜덤으로 나오게 설정하였다.

# 설계 – 코인 시스템

- 미사일과 마찬가지로 x값과 y값을 지정해주고 그 안에서 랜덤으로 나오게 설정하였다.

- 만약 비행기와 코인이 충돌하면 lblScore라고 이름을 정해준 label에 출력이 되게 하였다.

```
int score = 0;
참조 1개
void Coin(int speed)
{
    if (coin1.Top >= 800)
    {
        x = random.Next(0, 135);
        coin1.Location = new Point(x, 0);
    }
    else if (airplaneImg.Bounds.IntersectsWith(coin1.Bounds))
    {
        x = random.Next(0, 135);
        coin1.Location = new Point(x, 0);
        score++;
        lblScore.Text = score.ToString();
    }
    else
    {
        coin1.Top += speed;
    }

    if (coin2.Top >= 800)
    {
        x = random.Next(135, 270);
        coin2.Location = new Point(x, 0);
    }
    else if (airplaneImg.Bounds.IntersectsWith(coin2.Bounds))
    {
        x = random.Next(135, 270);
        coin2.Location = new Point(x, 0);
        score++;
        lblScore.Text = score.ToString();
    }
    else
    {
        coin2.Top += speed;
    }
```

# 설계 – 게임 오버

- 이 부분은 어려운 부분이라 블로그에 있는 코드를 그대로 따라했다.

- 비행기 이미지와 미사일들이 충돌하면 타이머의 활성화는 false가 되어 타이머에 있는 함수들이 멈추게 되고, Visible이 false인 GAME OVER가 써져 있는 label의 Visible이 true가 되어 보이게 구현했다.

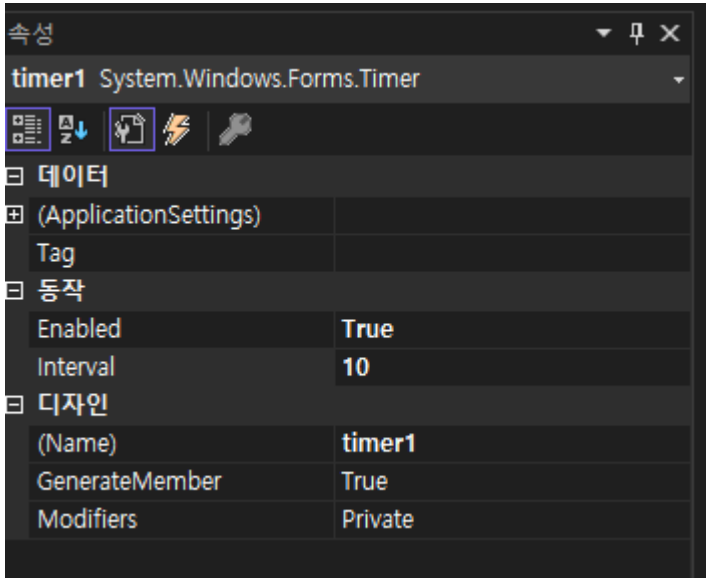
```
void GameOver()
{
    if (airplaneImg.Bounds.IntersectsWith(missile1.Bounds))
    {
        timer1.Enabled = false;
        lblGameOver.Visible = true;

    }
    else if (airplaneImg.Bounds.IntersectsWith(missile2.Bounds))
    {
        timer1.Enabled = false;
        lblGameOver.Visible = true;
    }
    else if (airplaneImg.Bounds.IntersectsWith(missile3.Bounds))
    {
        timer1.Enabled = false;
        lblGameOver.Visible = true;
    }
    else if (airplaneImg.Bounds.IntersectsWith(missile4.Bounds))
    {
        timer1.Enabled = false;
        lblGameOver.Visible = true;
    }
    else if (airplaneImg.Bounds.IntersectsWith(missile5.Bounds))
    {
        timer1.Enabled = false;
        lblGameOver.Visible = true;
    }
}
```

# 설계 - 타이머

- 타이머를 추가한 뒤 Enabled를 True로 바꿔 활성화 하고, Interval은 10밀리초(0.01초)로 설정하였다.

- 코드로 이동해 앞선 함수들을 호출하여 실행하게 했다.




```
private void timer1_Tick(object sender, EventArgs e)
{
    movecloud(5);
    Missile(8);
    Coin(4);
    MyMissile(4);
    GameOver();
}
```

# 실행