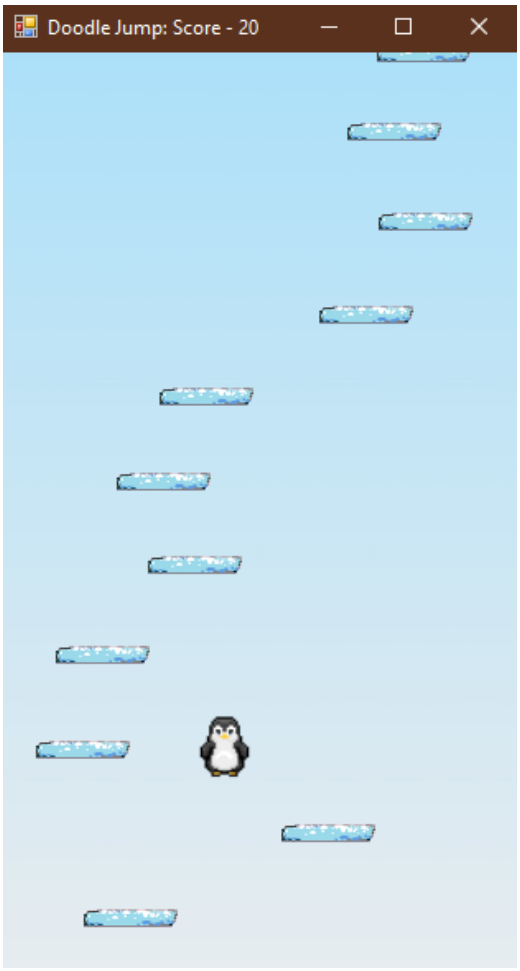
Documentation

For our project, we created a game based on the example of the game DoodleJump. In this game, you play as a penguin who jumps on ice platforms. Your goal is to score as many points as possible. On your way, you can come across both bonuses (such as a springboard and a jetpack) and monsters that you need to kill by shooting snowballs.



First of all, we created a public class Transform. Here we store the coordinates (position) and size of an object:

```
public class Transform
    {
        public PointF position;
        public Size size;

        public Transform(PointF position, Size size)
        {
            this.position = position;
            this.size = size;
        }
```

```
    }
```

The next step was to create a public class Platform. This class is directly responsible for platforms in our game:

```
public class Platform
    {
        Image sprite;
        public Transform transform;
        public int sizeX;
        public int sizeY;
        public bool isTouchedByPlayer;

        public Platform(PointF pos)
        {
            sprite = Properties.Resources.platform;
            sizeX = 60;
            sizeY = 12;
            transform = new Transform(pos, new Size(sizeX,
sizeY));
            isTouchedByPlayer = false;
        }

        public void DrawSprite(Graphics g)
        {
            g.DrawImage(sprite, transform.position.X,
transform.position.Y, transform.size.Width,
transform.size.Height);
        }
    }
```

Then we created a public static class PlatformController. With this class we will manipulate our platforms (create new ones or remove old ones):

```
public static class PlatformController
 {
        public static List<Platform> platforms;
        public static int startPlatformPosY = 400;
        public static int score = 0;

        public static void AddPlatform(PointF position)
        {
            Platform platform = new Platform(position);
            platforms.Add(platform);
        }

        public static void GenerateStartSequence()
        {
```

```
            Random r = new Random();
            for (int i = 0; i < 10; i++)
            {
                int x = r.Next(0, 270);
                int y = r.Next(50, 60);
                startPlatformPosY -= y;
                PointF position = new PointF(x,
startPlatformPosY);
                Platform platform = new Platform(position);
                platforms.Add(platform);
            }
        }

        public static void GenerateRandomPlatform()
        {
            ClearPlatforms();
            Random r = new Random();
            int x = r.Next(0, 270);
            PointF position = new PointF(x, startPlatformPosY);
            Platform platform = new Platform(position);
            platforms.Add(platform);
        }

        public static void ClearPlatforms()
        {
            for (int i = 0; i < platforms.Count; i++)
            {
                if (platforms[i].transform.position.Y >= 700)
                    platforms.RemoveAt(i);
            }
        }
    }
```

After that we created another class - public class Physics. With the help of this class, we implemented the physics of our game (i.e. our penguin jump and collide with the platforms on the map):

```
{
        public Transform transform;
        float gravity;
        float a;

        public float dx;

        public Physics(PointF position, Size size)
        {
            transform = new Transform(position, size);
            gravity = 0;
```

```
            a = 0.4f;
            dx = 0;
        }

        public void ApplyPhysics()
        {
            CalculatePhysics();
        }

        public void CalculatePhysics()
        {
            if (dx != 0)
            {
                transform.position.X += dx;
            }
            if (transform.position.Y < 700)
            {
                transform.position.Y += gravity;
                gravity += a;

                if (gravity > -25 && usedBonus)
                {
                    PlatformController.GenerateRandomPlatform();
                    PlatformController.startPlatformPosY = -200;
                    PlatformController.GenerateStartSequence();
                    PlatformController.startPlatformPosY = 0;
                    usedBonus = false;
                }

                Collide();
            }
        }

        public void Collide()
        {
            for (int i = 0; i <
PlatformController.platforms.Count; i++)
            {
                var platform = PlatformController.platforms[i];
                if (transform.position.X+transform.size.Width/2 >=
platform.transform.position.X && transform.position.X +
transform.size.Width/2 <= platform.transform.position.X +
platform.transform.size.Width)
                {
                    if (transform.position.Y+transform.size.Height
>= platform.transform.position.Y && transform.position.Y +
transform.size.Height <= platform.transform.position.Y +
platform.transform.size.Height)
```

```
                    {
                        if (gravity > 0)
                        {
                            AddForce();
                            if (!platform.isTouchedByPlayer)
                            {
                                PlatformController.score += 20;

PlatformController.GenerateRandomPlatform();
                                platform.isTouchedByPlayer = true;
                            }
                        }
                    }
                }
            }
        }

        public void AddForce(int force = -10)
        {
            gravity = force;
        }
    }
```

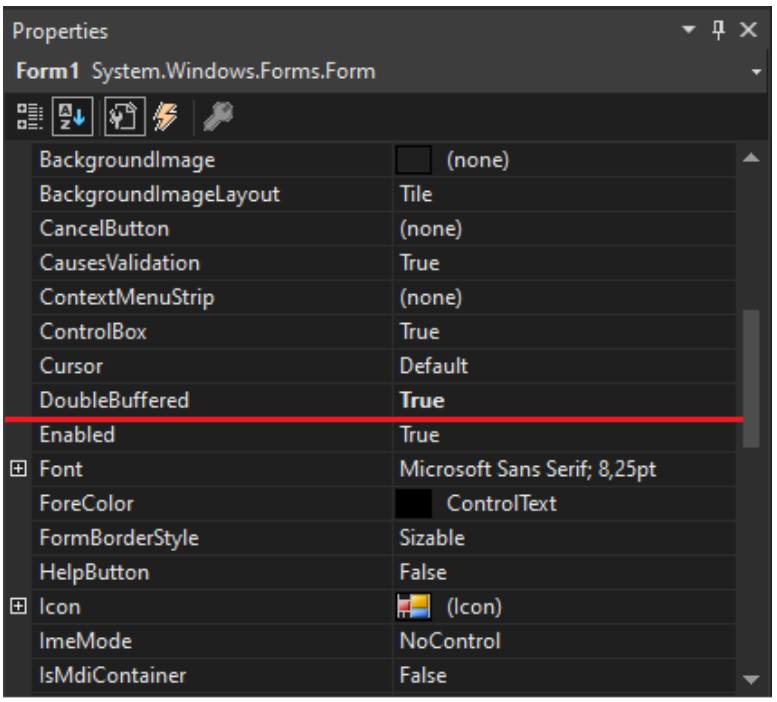
The next task was to write the public class Player, which will be responsible for our player:

```
public class Player
    {
        public Physics physics;
        public Image sprite;

        public Player()
        {
            sprite = Properties.Resources.penguin;
            physics = new Physics(new PointF(100, 350), new
Size(36, 47));
        }

        public void DrawSprite(Graphics g)
        {
            g.DrawImage(sprite, physics.transform.position.X,
physics.transform.position.Y, physics.transform.size.Width,
physics.transform.size.Height);
        }
    }
```

For smooth rendering of all characters, we have enabled double buffering:



After all this, we added the public class Bullet:

```
public class Bullet
    {
        public Physics physics;
        public Image sprite;

        public Bullet(PointF pos)
        {
            sprite = Properties.Resources.bullet;
            physics = new Physics(pos, new Size(15, 15));
        }

        public void MoveUp()
        {
            physics.transform.position.Y -= 15;
        }

        public void DrawSprite(Graphics g)
        {
            g.DrawImage(sprite, physics.transform.position.X,
physics.transform.position.Y, physics.transform.size.Width,
physics.transform.size.Height);
        }
    }
```

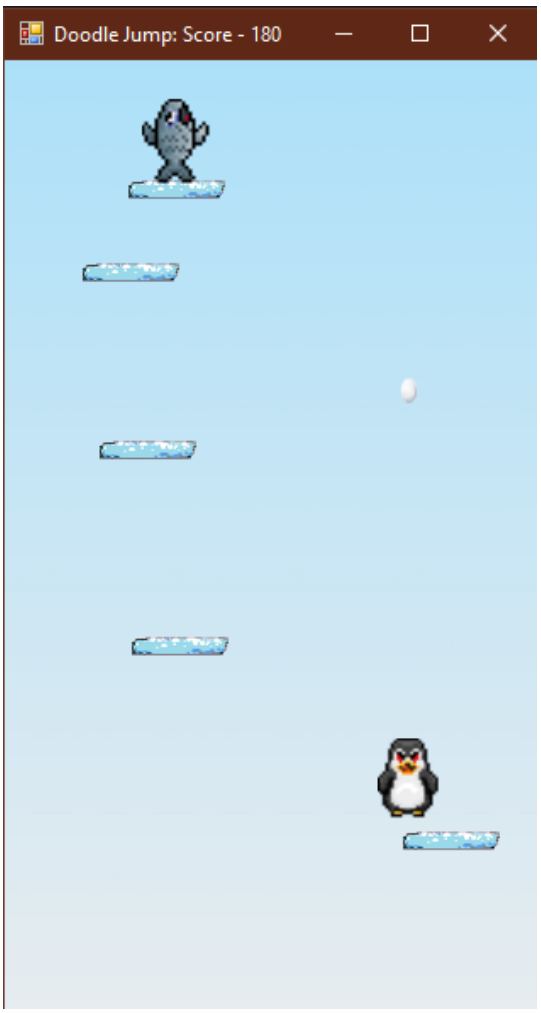
The next goal was to create monsters and bonuses to make our game more interesting:

```
public class Enemy : Player
```

```
{
    public Enemy(PointF pos, int type)
    {
        switch (type)
        {
            case 1:
                sprite = Properties.Resources.enemy1r;
                physics = new Physics(pos, new Size(40, 50));
                break;
            case 2:
                sprite = Properties.Resources.enemy2r;
                physics = new Physics(pos, new Size(50, 24));
                break;
            case 3:
                sprite = Properties.Resources.enemy3r;
                physics = new Physics(pos, new Size(60, 64));
                break;

        }

    }
}
```



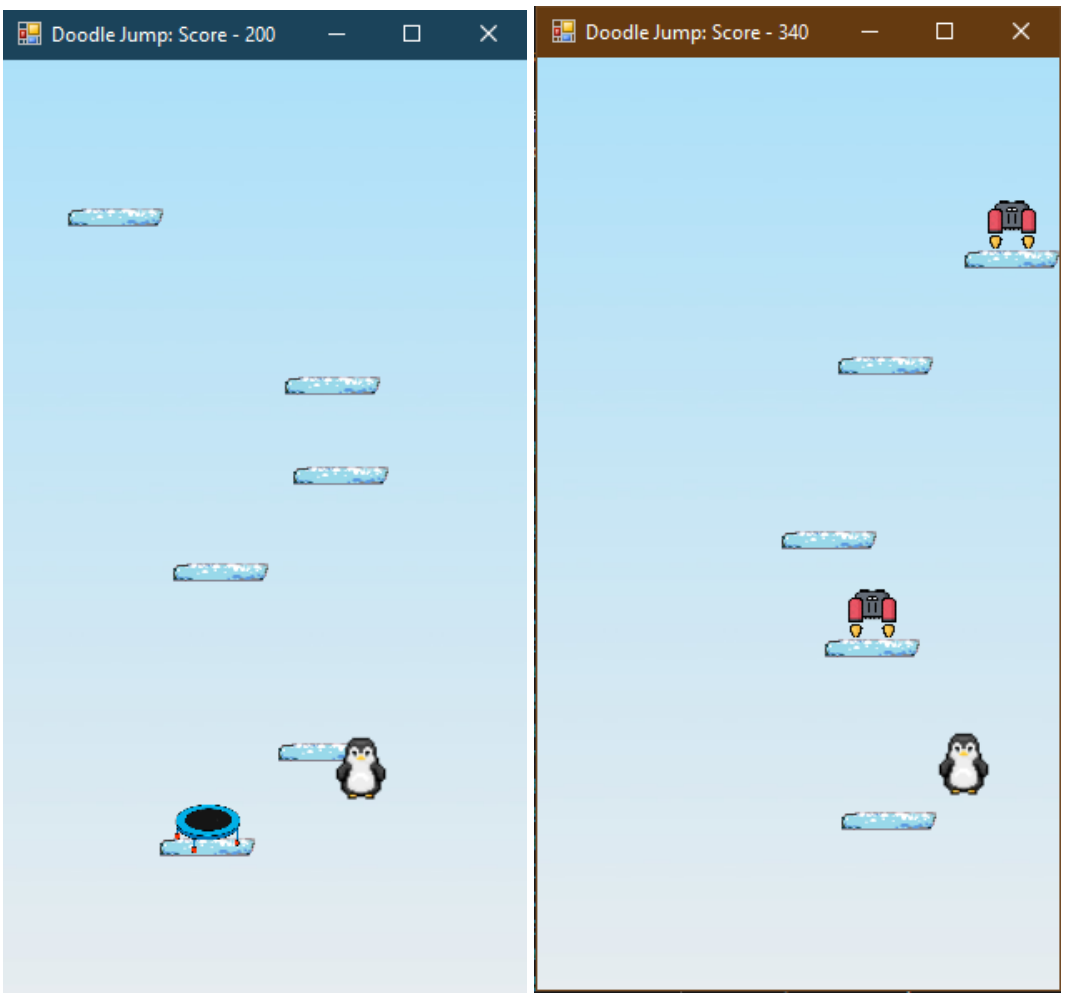

```
public class Bonus
    {

        public Physics physics;
        public Image sprite;
        public int type;

        public Bonus(PointF pos, int type)
        {
            switch (type)
            {
                case 1:
                    sprite = Properties.Resources.spring;
                    physics = new Physics(pos, new Size(40, 30));
                    break;
                case 2:
                    sprite = Properties.Resources.jetpack;
                    physics = new Physics(pos, new Size(30, 30));
                    break;
            }
            this.type = type;
```

```
        }

        public void DrawSprite(Graphics g)
        {
            g.DrawImage(sprite, physics.transform.position.X,
physics.transform.position.Y, physics.transform.size.Width,
physics.transform.size.Height);
        }
    }
```



And the final part of the code:

```
public partial class Form1 : Form
    {

        Player player;
        Timer timer1;
        public Form1()
        {
            InitializeComponent();
```

```
            Init();
            timer1 = new Timer();
            timer1.Interval = 15;
            timer1.Tick += new EventHandler(Update);
            timer1.Start();
            this.KeyDown += new
KeyEventHandler(OnKeyboardPressed);
            this.KeyUp += new KeyEventHandler(OnKeyboardUp);
            this.BackgroundImage = Properties.Resources.back;
            this.Height = 600;
            this.Width = 330;
            this.Paint += new PaintEventHandler(OnRepaint);
        }

        public void Init()
        {
            PlatformController.platforms = new
System.Collections.Generic.List<Platform>();
            PlatformController.AddPlatform(new
System.Drawing.PointF(100, 400)); //create a starting platform
under the character
            PlatformController.startPlatformPosY = 400;
            PlatformController.score = 0;
            PlatformController.GenerateStartSequence();
            PlatformController.bullets.Clear();
            PlatformController.bonuses.Clear();
            PlatformController.enemies.Clear();
            player = new Player();
        }

        private void OnKeyboardUp(object sender, KeyEventArgs e)
        {
            player.physics.dx = 0;
            player.sprite = Properties.Resources.penguin;
            switch (e.KeyCode.ToString())
            {
                case "Space":
                    PlatformController.CreateBullet(new
PointF(player.physics.transform.position.X +
player.physics.transform.size.Width / 2,
player.physics.transform.position.Y));
                    break;
            }
        }

        private void OnKeyboardPressed(object sender, KeyEventArgs
e)
        {
```

```
            switch (e.KeyCode.ToString())
            {
                case "Right":
                    player.physics.dx = 6;
                    break;
                case "Left":
                    player.physics.dx = -6;
                    break;
                case "Space":
                    player.sprite =
Properties.Resources.s_penguin;
                    break;
            }
        }


        private void Update(object sender, EventArgs e)
        {
            this.Text = "Doodle Jump: Score - " +
PlatformController.score;

            if ((player.physics.transform.position.Y >=
PlatformController.platforms[0].transform.position.Y + 200) ||
player.physics.StandartCollidePlayerWithObjects(true, false))
                Init();

            player.physics.StandartCollidePlayerWithObjects(false,
true);

            if (PlatformController.bullets.Count > 0)
            {
                for (int i = 0; i <
PlatformController.bullets.Count; i++)
                {
                    if
(Math.Abs(PlatformController.bullets[i].physics.transform.position
.Y - player.physics.transform.position.Y) > 500)
                    {
                        PlatformController.RemoveBullet(i);
                        continue;
                    }
                    PlatformController.bullets[i].MoveUp();
                }
            }
            if (PlatformController.enemies.Count > 0)
            {
                for (int i = 0; i <
PlatformController.enemies.Count; i++)
```

```
                {
                    if
(PlatformController.enemies[i].physics.StandartCollide())
                    {
                        PlatformController.RemoveEnemy(i);
                        break;
                    }
                }
            }

            player.physics.ApplyPhysics();
            FollowPlayer();

            Invalidate();
        }

        public void FollowPlayer()
        {
            int offset = 400 -
(int)player.physics.transform.position.Y;
            player.physics.transform.position.Y += offset;
            for (int i = 0; i <
PlatformController.platforms.Count; i++)
            {
                var platform = PlatformController.platforms[i];
                platform.transform.position.Y += offset;
            }
            for (int i = 0; i < PlatformController.bullets.Count;
i++)
            {
                var bullet = PlatformController.bullets[i];
                bullet.physics.transform.position.Y += offset;
            }
            for (int i = 0; i < PlatformController.enemies.Count;
i++)
            {
                var enemy = PlatformController.enemies[i];
                enemy.physics.transform.position.Y += offset;
            }
            for (int i = 0; i < PlatformController.bonuses.Count;
i++)
            {
                var bonus = PlatformController.bonuses[i];
                bonus.physics.transform.position.Y += offset;
            }
        }

        private void OnRepaint(object sender, PaintEventArgs e)
```

```
        {
            Graphics g = e.Graphics;
            if (PlatformController.platforms.Count > 0)
            {
                for (int i = 0; i <
PlatformController.platforms.Count; i++)
                    PlatformController.platforms[i].DrawSprite(g);
            }
            if (PlatformController.bullets.Count > 0)
            {
                for (int i = 0; i <
PlatformController.bullets.Count; i++)
                    PlatformController.bullets[i].DrawSprite(g);
            }
            if (PlatformController.enemies.Count > 0)
            {
                for (int i = 0; i <
PlatformController.enemies.Count; i++)
                    PlatformController.enemies[i].DrawSprite(g);
            }
            if (PlatformController.bonuses.Count > 0)
            {
                for (int i = 0; i <
PlatformController.bonuses.Count; i++)
                    PlatformController.bonuses[i].DrawSprite(g);
            }
            player.DrawSprite(g);
        }
    }
```