SPACE BOMBER

TOMASZ KAWKA MATEUSZ KMAK

ZARYS PROJEKTU

- Gierka na Androida
- Gameplay:
  - Wysadzanie w powietrze losowo poruszających się kulek
  - Punkty za wysadzanie dobrych kulek, utrata życia za wysadzanie złych
  - System power-upów (większe wybuchy, nieśmiertelność, itp.)
  - 2 tryby gry: do utraty wszystkich żyć i na czas
- Grafika 2D/3D
- Efekty dźwiękowe i fajna muzyczka
- (być może) Połączenie z Google Play (leaderboardy, achievementy)

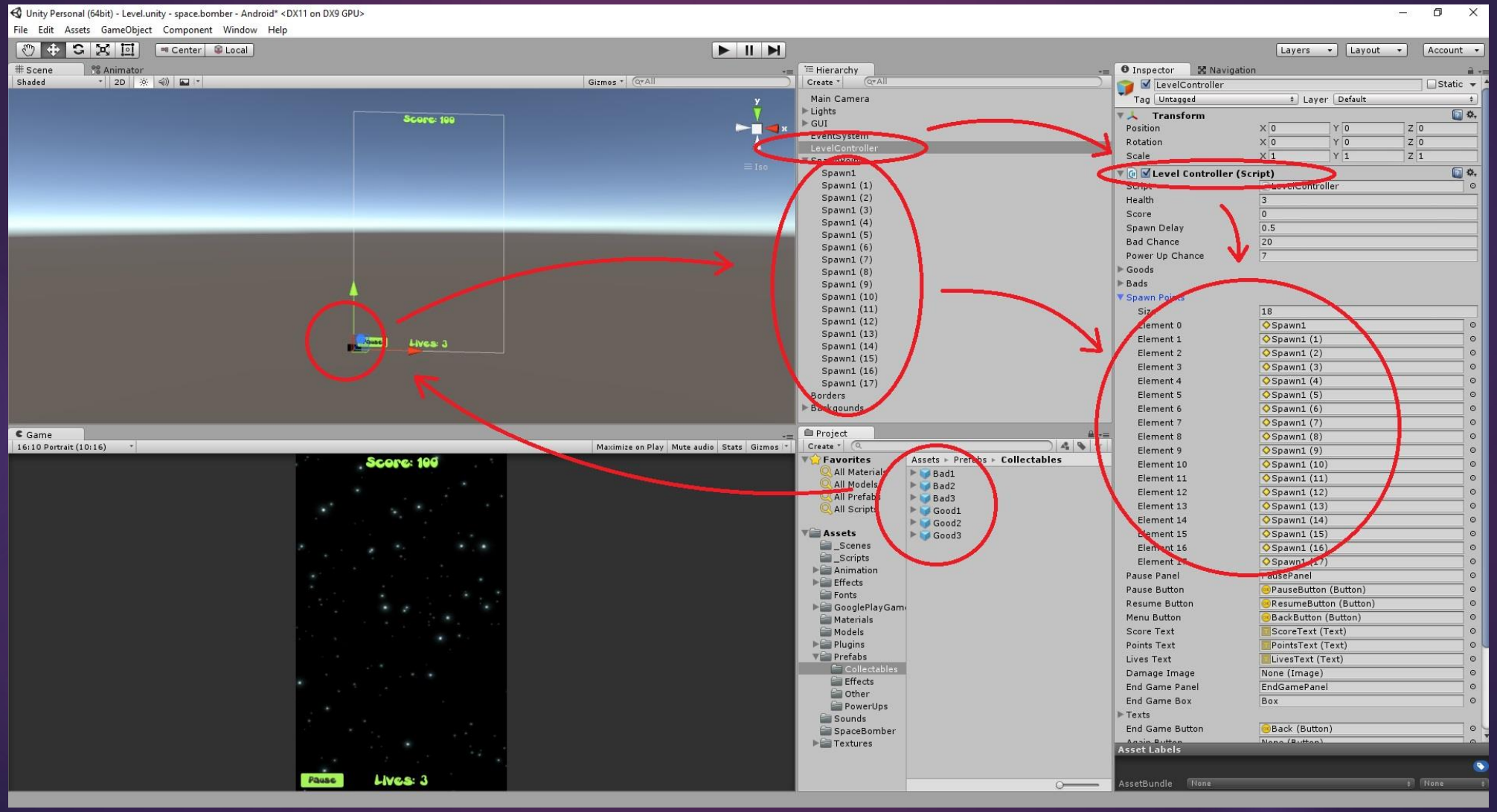
WYKORZYSTANE NARZĘDZIA

- Unity 3D + VS2015
- Wszystkie skrypty napisane od zera (C#)
- Darmowe assety graficzne/dźwiękowe/fx z Unity Asset Store
- Tekstury własnej roboty (MS Paint)

STRUKTURA PROJEKTU

- Sceny → stany w skończonej maszynie stanów
  - Menu – scena startowa
  - Level – główna scena, w której odbywa się rozgrywka
- Kontrolery → globalne obiekty kontrolujące działanie gry
  - GameController – przejścia między scenami i ustawienia globalne
  - AudioController – odtwarzanie efektów dźwiękowych
  - MenuController – nawigacja po menu głównym (tylko scena Menu)
  - LevelController – ogólna logika całej gry (tylko scena Level)

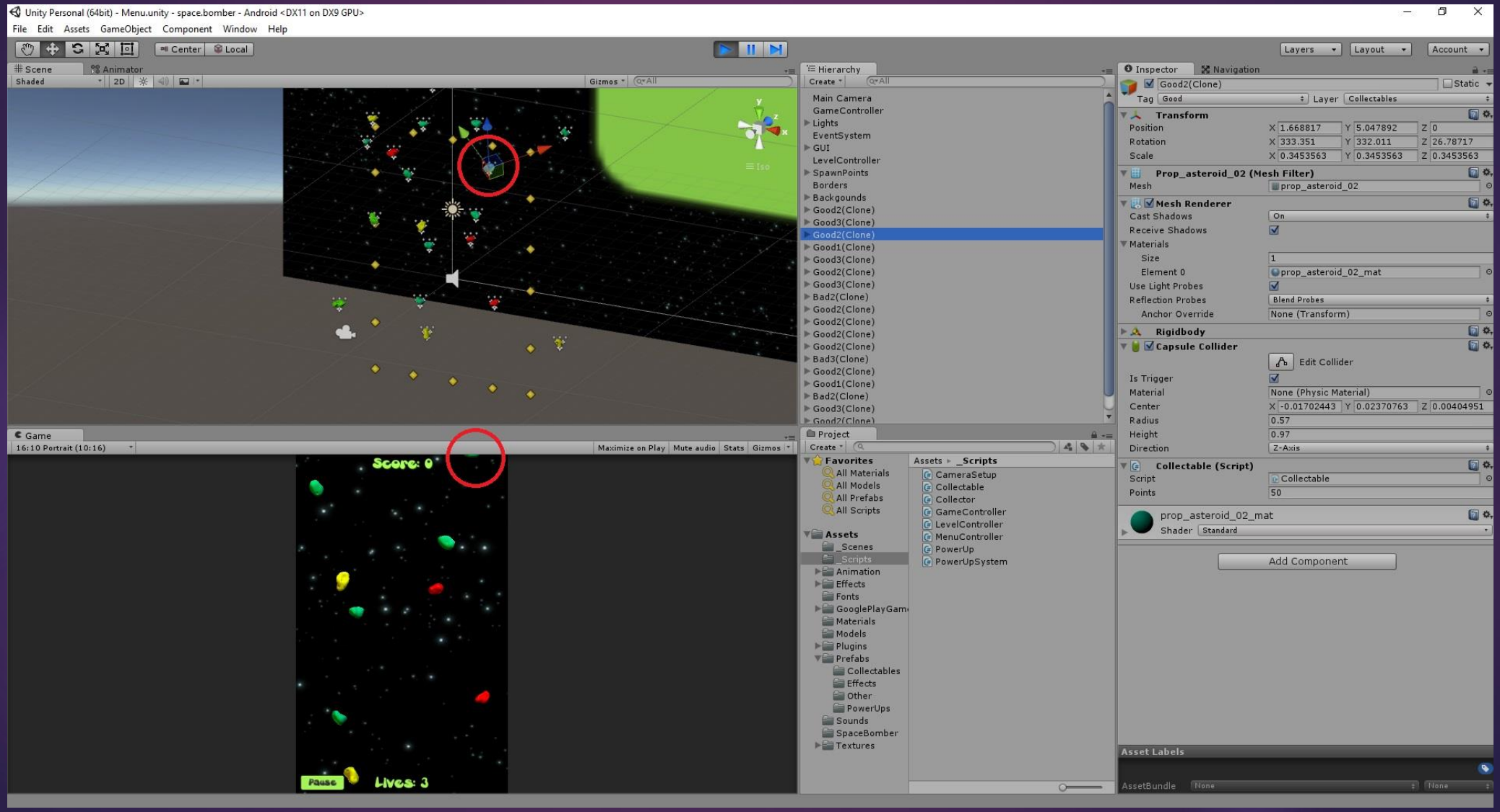
LOGIKA GRY


Unity Personal (64bit) - Level.unity - space.bomber - Android* <DX11 on DX9 GPU>
File Edit Assets GameObject Component Window Help
Hierarchy
Main Camera
Lights
GUI
EventSystem
LevelController
Spawn1
Spawn1 (1)
Spawn1 (2)
Spawn1 (3)
Spawn1 (4)
Spawn1 (5)
Spawn1 (6)
Spawn1 (7)
Spawn1 (8)
Spawn1 (9)
Spawn1 (10)
Spawn1 (11)
Spawn1 (12)
Spawn1 (13)
Spawn1 (14)
Spawn1 (15)
Spawn1 (16)
Spawn1 (17)
Borders
Backgounds
Inspector
LevelController
Tag Untagged
Layer Default
Transform
Position
Rotation
Scale
Level Controller (Script)
Health 3
Score 0
Spawn Delay 0.5
Bad Chance 20
Power Up Chance 7
Goods
Bads
Spawn Points
Size 18
Element 0 Spawn1
Element 1 Spawn1 (1)
Element 2 Spawn1 (2)
Element 3 Spawn1 (3)
Element 4 Spawn1 (4)
Element 5 Spawn1 (5)
Element 6 Spawn1 (6)
Element 7 Spawn1 (7)
Element 8 Spawn1 (8)
Element 9 Spawn1 (9)
Element 10 Spawn1 (10)
Element 11 Spawn1 (11)
Element 12 Spawn1 (12)
Element 13 Spawn1 (13)
Element 14 Spawn1 (14)
Element 15 Spawn1 (15)
Element 16 Spawn1 (16)
Element 17 Spawn1 (17)
Pause Panel PausePanel
Pause Button PauseButton (Button)
Resume Button ResumeButton (Button)
Menu Button BackButton (Button)
Score Text ScoreText (Text)
Points Text PointsText (Text)
Lives Text LivesText (Text)
Damage Image None (Image)
End Game Panel EndGamePanel
End Game Box Box
Texts
End Game Button Back (Button)
Score: 100
Lives: 3
Pause
Project
Favorites
Assets
_Scenes
_Scripts
Animation
Effects
Fonts
Materials
Models
Plugins
Prefabs
Collectables
Effects
Other
PowerUps
Sounds
SpaceBomber
Textures
Bad1
Bad2
Bad3
Good1
Good2
Good3
Asset Labels
AssetBundle

LOGIKA GRY C.D.

- LevelController jest obiektem klasy LevelController : MonoBehaviour, dostępnym globalnie ze sceny Level
- LevelController ma propertę GameObject[] SpawnPoints
- Na scenie ustawionych jest n obiektów GameObject reprezentujących punkty, z których wylatują kulki; SpawnPoints trzyma referencje na te obiekty
- Kulki nie są od razu na scenie, mają postać prefabów (obiektów jakiejś klasy, w tym przypadku Collectable : MonoBehaviour), zostają ustawiane w czasie rzeczywistym przez LevelController

LOGIKA GRY C.D.



LOGIKA GRY C.D.

- Po odpaleniu gry w pozycji losowego elementu LevelController.spawnPoints z opóźnieniem LevelController.spawnDelay zostają utworzone nowe instancje jednego z sześciu typów kulek

```
void Update()
{
    if (!gameOver)
    {
        spawnCounter += Time.deltaTime;

        if (spawnCounter >= spawnDelay)
        {
            spawnCounter = 0f;
            int spawnPoint = Random.Range(0, 18);
            int collectable = Random.Range(0, 3);
            int spawnRoll = Random.Range(0, 100);

            if (spawnRoll >= badChance)
            {
                Collectable col = (Collectable)Instantiate(goods[collectable], spawnPoints[spawnPoint].transform.position, Quaternion.identity);
                switch(collectable)
                {
                    case 0:
                        col.color = "Green";
                        break;
                    case 1:
                        col.color = "Cyan";
                        break;
                    case 2:
                        col.color = "Yellow";
                        break;
                }
            }
            else
            {
                Collectable col = (Collectable)Instantiate(bads[collectable], spawnPoints[spawnPoint].transform.position, Quaternion.identity);
                col.color = "Red";
            }

        }
    }
}
```

LOGIKA GRY C.D.

- Po utworzeniu instancji dowolnego obiektu dziedziczącego po MonoBehaviour wywoływana jest jego metoda Awake()
- Dla Collectable losujemy wielkość oraz parametry fizyczne (prędkość kątową, prędkość)

```
protected virtual void Awake()
{
    float scale = Random.Range(-0.05f, 0.05f);
    transform.localScale += new Vector3(scale, scale, scale);

    effect = GetComponentInChildren<ParticleSystem>();

    rb = GetComponent<Rigidbody>();

    rb.angularVelocity = new Vector3(Random.Range(-90f, 90f), Random.Range(-90f, 90f), Random.Range(-90f, 90f));

    float destinationX = Random.Range(-1.5f, 1.5f);
    float destinationY = Random.Range(-2.0f, 4.0f);

    destination = new Vector3(destinationX, destinationY, 0f);

    speed = Random.Range(0.7f, 3.0f);

    Vector3 velocityVector = destination - transform.position;
    velocityVector.Normalize();

    rb.velocity = velocityVector * speed;
}
```

LOGIKA GRY C.D.

- Po tym jak kulki opuszczą widoczną część sceny należy je usunąć (dbamy o wydajność gierki)
- Widoczna część sceny objęta jest niewidocznym dla gracza obiektem typu Collider (Box Collider), otagowanym jako „Border”
- Każdy obiekt Collectable również posiada własny komponent Collider
- Collider naszej kulki po wyjściu z obszaru innego Collidera wywołuje metodę OnTriggerExit, której argumentem jest drugi Collider
- Jeśli Collider z którego wydostała się kulka jest otagowany jako „Border”, kulka zostaje usunięta ze sceny

```
protected void OnTriggerExit(Collider other)
{
    if (other.tag == "Border")
    {
        GameObject.Find("LevelController").GetComponent<LevelController>().missedItems += 1;
        Destroy(gameObject);
    }
}
```

CZEGO SIĘ NAUCZYLIŚMY

- Obsługi Unity 3D:
  - Sceny 3D (obiekty w 3D, grafika, kamera), animacja, oświetlenie, GUI, fizyka ciała stałego, efekty specjalne, dźwięk
- Wzorców projektowych:
  - Skończona maszyna stanów (sceny, animacje)
  - Singleton (GameController)