WYŻSZA SZKOŁA
INFORMATYKI i ZARZĄDZANIA
z siedzibą w Rzeszowie

Dokumentacja projektu

Przedmiot: **Wzorce projektowe i architektura aplikacji**

Tytuł projektu: **Gra logiczna „Tap color objects”**

Prowadzący:

Dr Marek Jaszuk

Wykonawca: **Mateusz Molik** - **65304**

Semestr, symbol kierunku i grupa: **I / INF / SPO2**

Rzeszów, 2021

- # Opis projektu

Gra mobilna w której to gracz ma za zadanie kliknąć w odpowiednie obiekty o danym kolorze. Gracz wygrywa jeżeli wszystkie obiekty o danym kolorze zostaną kliknięte. Jeżeli gracz kliknie zły obiekt to przegrywa. Celem gracza jest pobijanie własnego rekordu.

- # Wykorzystane wzorce

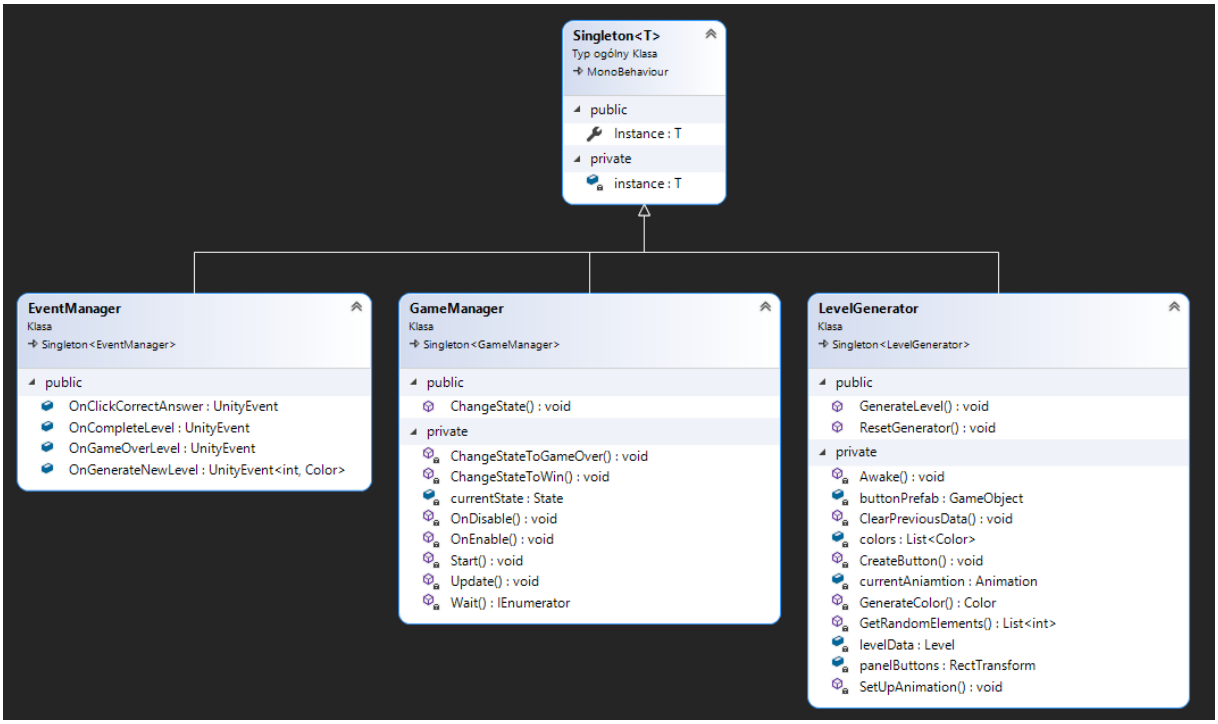
W projekcie zostały wykorzystane następujące wzorce:

- **Singleton** (GameManager, LevelGenerator, PanelManager oraz EventManager)
- **Stan** (State, MenuState, GameState, GameOverState, WinState)
- **Most** (Animation, ScaleAnimation, FadeAnimation, SpecialAnimation)
- **Obserwator** (eventy: OnGenerateNewLevel, OnClickCorrectAnswer, OnCompleteLevel, OnGameOverLevel)
- **Fabryka (metoda wytwórcza)** (FactoryButton, FactorySquareButton, FactoryCircleButton / Button, SquareButton, CircleButton)

- # Szczegółowy opis wykorzystanych wzorców

- # Singleton

Singleton został wykorzystany w kilku klasach w projekcie gdzie instancje tych klas są wykorzystane z różnego miejsca w projekcie. W projekcie użyto kilkukrotnie wzorca singleton, dlatego została stworzona klasa generyczna **Singleton**. Diagram klas przedstawiający wykorzystanie wzorca singleton w projekcie.



Implementacja w C#:

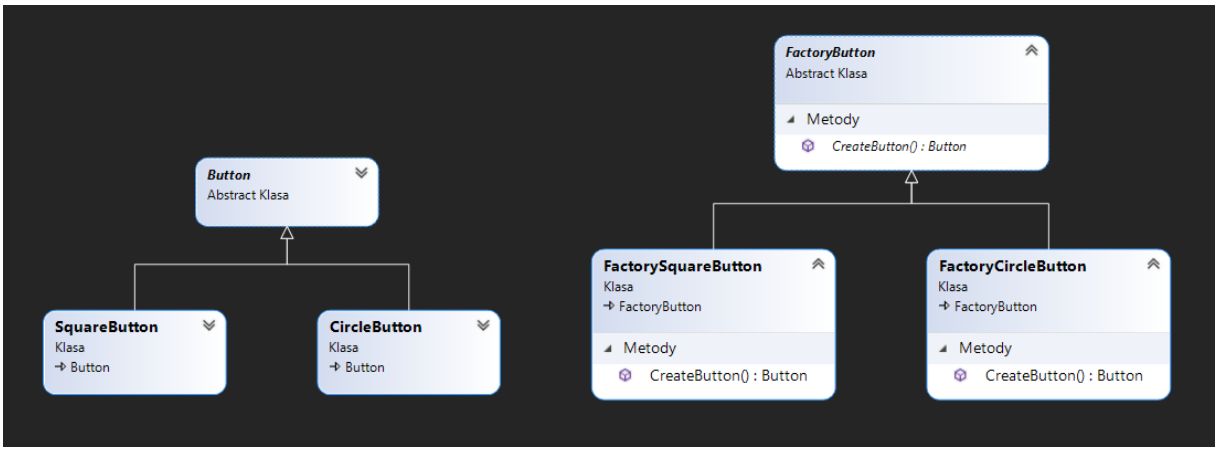
```
Skrypt aparatu Unity | Odwołania: 5
public class Singleton<T> : MonoBehaviour where T : MonoBehaviour
{
    private static T instance;
    Odwołania: 26
    public static T Instance
    {
        get
        {
            if (instance == null)
            {
                instance = FindObjectOfType<T>();
            }
            return instance;
        }

    }
}
```

```
Skrypt aparatu Unity | Odwołania: 19
public class EventManager : Singleton<EventManager>
{
```

- # Fabryka - metoda wytwórcza

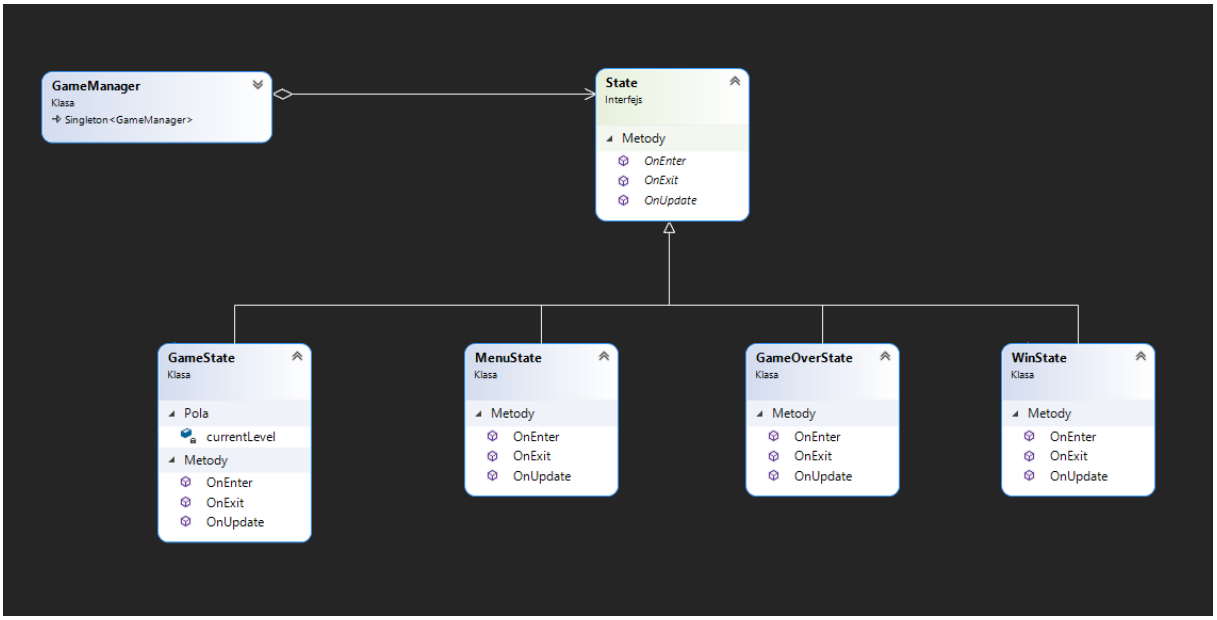
Wzorzec fabryka został wykorzystany do procesu tworzenia przycisków interaktywnych na scenie. Na samym początku gry, gracz ma możliwość wyboru jakiego kształtu będą przyciski na ekranie.



- # Stan (Maszyna stanów)

Wzorzec stan został wykorzystany do zarzadzania logiką gry. Należy zwrócić że gra oferuje tylko cztery stany w jakich gra może się znajdować, są to:

- **Menu State** - gracz znajduję się w menu gry
- **Game State** - gracz toczy rozgrywkę
- **GameOver State** - gracz przegrał
- **Win State** - gracz wygrał



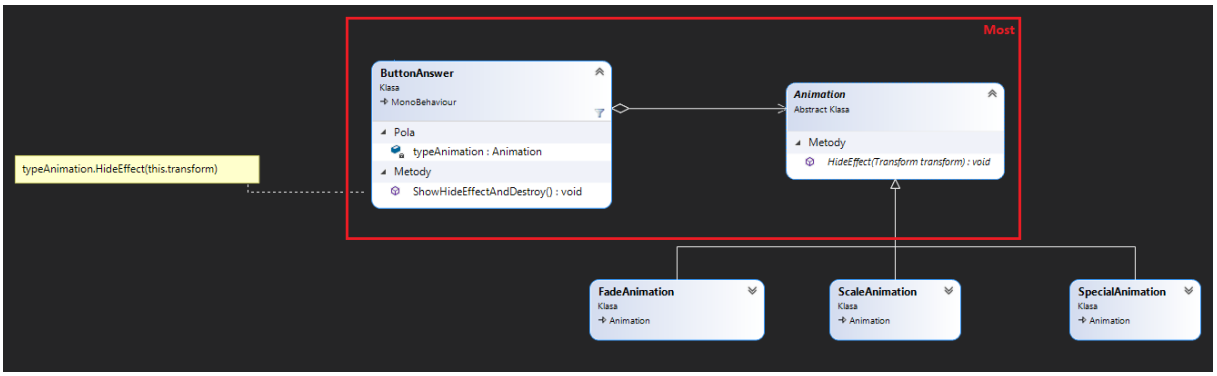
Zmiana stanów gry odbywa się za pomocą klasy **GameManager**.

```
Assembly-CSharp
23      }
24
        Odwołania: 5
25      public void ChangeState(State newState)
26      {
27          if(currentState != null)
28          {
29              currentState.OnExit();
30          }
31
32          currentState = newState;
33
34          currentState.OnEnter();
35      }
36
```

```
Odwołania: 6
public interface State
{
    Odwołania: 5
    public void OnEnter();
    Odwołania: 5
    public void OnUpdate();
    Odwołania: 5
    public void OnExit();
}
```

- **Most**

Wzorzec most został wykorzystany do podmiany typów animacji przycisków. Za każdym razem kiedy gracz kliknie poprawny przycisk, nastąpi unikatowa animacja. Dzięki zastosowaniu wzorca most istnieje możliwość ustawianie różnych animacji dla przycisków.



Implantacja w c#:

```
Odwołania: 6
public abstract class Animation
{
    Odwołania: 4
    public abstract void HideEffect(Transform transform);
}

1 odwołanie
public class ScaleAnimation : Animation
{
    Odwołania: 4
    public override void HideEffect(Transform transform)
    {
        transform.GetComponent<RectTransform>().DOScale(Vector3.zero, 1f);
    }
}

1 odwołanie
public class FadeAnimation : Animation
{
    Odwołania: 4
    public override void HideEffect(Transform transform)
    {
        transform.GetComponent<Image>().DOFade(0, 1f);
    }
}

Odwołania: 2
public class SpecialAnimation : Animation
{
    Odwołania: 4
    public override void HideEffect(Transform transform)
    {
        transform.GetComponent<RectTransform>().DORotate(new Vector3(0,0, 90), 1f);
        transform.GetComponent<RectTransform>().DOScale(Vector3.zero, 1f);
    }
}
```

- # Obserwator

Do zrealizowania wzorca obserwator został wykorzystany system do zarządzania eventami z gotowej biblioteki tj. (**UnityEngine.Events**). W projekcie stworzono 4 eventy:

- **OnClickCorrectAnswer** – Event który zostaje odpalony gdy gracz kliknie poprawną odpowiedź.
- **OnGenerateNewLevel** – Event który zostaje odpalony gdy zostanie stworzony nowy level.
- **OnCompleteLevel** – Event który zostaje odpalony gry przejdzie level.
- **OnGameOverLevel** – Event który zostaje odpalony gry gracz kliknie nie poprawą odpowiedz .

```
Skrypt aparatu Unity | Odwołania: 19
public class EventManager : Singleton<EventManager>
{
    public UnityEvent OnClickCorrectAnswer;
    public UnityEvent<int, Color> OnGenerateNewLevel;
    public UnityEvent OnCompleteLevel;
    public UnityEvent OnGameOverLevel;
}
```

Przykład działania eventa OnGenerateNewLevel.

Do eventa **OnGenerateNewLevel** podpięty jest skrypt który opowiada za wyświetlanie informacji o obecnym poziomie w którym gracz się znajduje tj. **LevelText.** Za każdym razem kiedy zostanie wygenerowany nowy poziom, event **OnGenerateNewLevel** zostanie odpalony i powiadomi wszystkie skrypty które go nasłuchują.

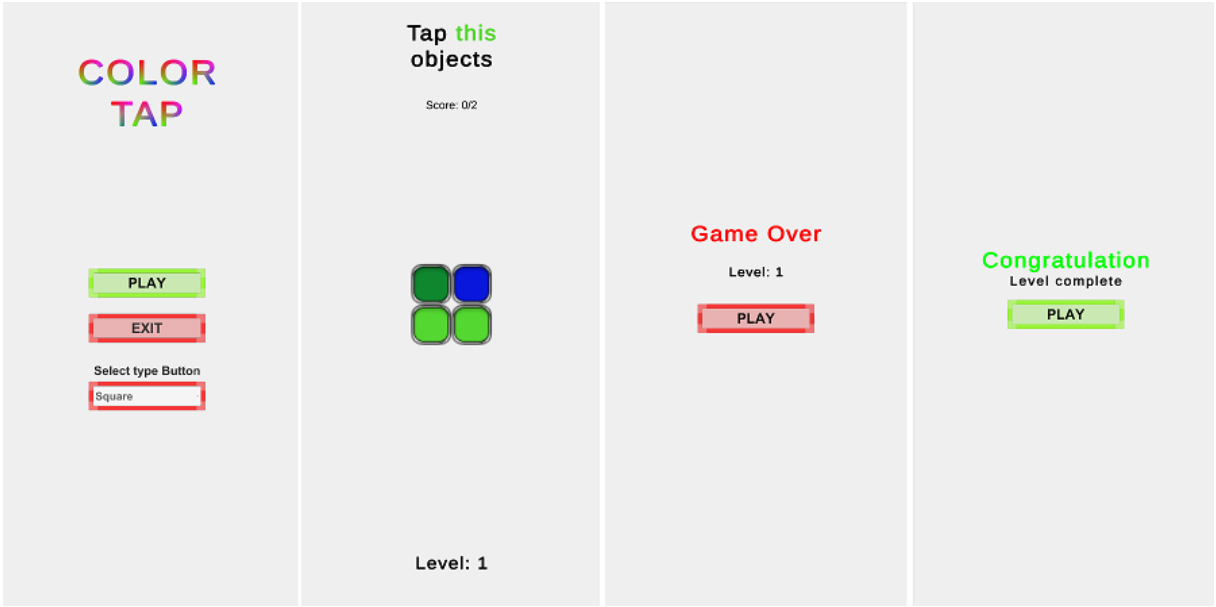
LevelText.cs

Assembly-CSharp

LevelText

```
5
    Skrypt aparatu Unity | Odwołania: 0
6   public class LevelText : MonoBehaviour
7   {
8       [SerializeField]
9       private TextMeshProUGUI text;
10      private int currentLevel = 0;
11
        Unity Message | Odwołania: 0
12      private void OnEnable()
13      {
14          EventManager.Instance.OnGenerateNewLevel.AddListener(UpdateText);
15          EventManager.Instance.OnGameOverLevel.AddListener(ResetLevel);
16      }
17
        Odwołania: 2
18      private void UpdateText(int score, Color color)
19      {
20          currentLevel++;
21          text.text = $"Level: {currentLevel}";
22      }
23
        Odwołania: 2
24      private void ResetLevel()
25      {
26          currentLevel = 0;
27      }
28
        Unity Message | Odwołania: 0
29      private void OnDisable()
30      {
31          EventManager.Instance.OnGenerateNewLevel.RemoveListener(UpdateText);
32          EventManager.Instance.OnGameOverLevel.RemoveListener(ResetLevel);
33      }
34  }
```

- **Widoki z gry**



- **Kod aplikacji**

Link do githuba: https://github.com/mix923/ZaliczenieWzorce