WYŻSZA SZKOŁA EKONOMII I INFORMATYKI

KATEDRA INFORMATYKI

Eva: The Beginning

Dokumentacja techniczna

Imię i nazwisko: Michał CHOWANIEC

January 4, 2025

Contents

List of Figures

List of Tables

1 Game Overview

"Eva: The Beginning" is a turn-based strategy game where players control a hybrid creature navigating a dynamic world. The game revolves around managing resources such as water, energy, biomass, and protein while participating in exploration and evolution. The goal is to make strategic decisions on movement, resource allocation, and world interaction, with an emphasis on survival, growth, and player customization.

Built using **Unity 6.000.0.27f1**, all scripts in the game are written in **C#**, leveraging Unity's extensive engine features to handle player interactions, world building and resource management. [**?**]

2 Core Systems and Components

2.1 Player System

2.1.1 Player

Each player is represented by a `Player` object (ScriptableObject) containing key attributes such as:

- `Water`, `Energy`, `Biomass`, `Protein`: Resources required for survival and evolution.
- `Pos` and `Rot`: Position and rotation of the player in the game world.
- `Prefab` and `TreePrefab`: Prefabs used to instantiate the player and their associated objects.
- `human`: Boolean flag to indicate whether the player is human- or AI-controlled.

```
1 public class Player : ScriptableObject
2 {
3     public int water, energy, biomass, protein;
4     public Vector3 Pos, StartPos;
5     public Quaternion Rot, StartRot;
6     public GameObject Prefab;
7     public GameObject TreePrefab;
8     public bool human;
9 }
```

2.1.2 PlayerManager

Manages multiple players, including allocating players to the world and managing active player transitions between turns. It invokes events when the active player changes.

```
public class PlayerManager : MonoBehaviour
{
    public Player[] players;
    private Player activePlayer;
    private int activePlayerIndex;

    public void AllocatePlayers() { ... }
    public void ChangePlayer() { ... }
}
```

2.1.3 PlayerMovement

Handles player movement in the world by moving the player's character from one location to another, based on player input. Utilizes Unity's coroutine system for smooth movement.

```
public class PlayerMovement : Movement
{
    // Additional movement logic here
}
```

2.1.4 PlayerScript

Controls individual player interactions, such as checking the player's position on tiles and initiating related actions, like movement or interaction with objects.

```
public class PlayerScript : MonoBehaviour
{
    public Player player;
    public Animator animator;

    public void CheckPosition() { ... }
}
```

2.2 Tile and Interaction Systems

2.2.1 TileType

Defines the properties of different tiles in the game. Each tile has a description, type (Rock, Grass, Sand, Water), and attributes like passability and rootability. Tiles are represented in the game world as GameObjects.

```
public class TileType : ScriptableObject
{
    public string Description;
    public TileTypesEnum type;
    public bool passable;
    public bool rootable;
    public Material Material;
}
```

2.2.2 TileTypesEnum

An enum that categorizes tiles into distinct types such as Rock, Grass, Sand, and Water, affecting how players interact with the world.

```
public enum TileTypesEnum
{
    Rock,
    Grass,
    Sand,
    Water
}
```

2.2.3 GameObjectTypeEnum

An enum that classifies objects that can be placed on tiles (e.g., Rock, Bush, Tree, Water).

```
public enum GameObjectTypeEnum
{
    Rock,
    Bush,
    Tree,
    Water,
```

```
7       None
8   }
```

2.2.4 OnClickTile

Handles tile selection by the player, triggering actions based on the selected tile's position when clicked. Uses Unity's event system for interaction.

```
1 public class OnClickTile : MonoBehaviour
2 {
3       public static event Action<Vector3> OnClick;
4
5       private void OnMouseDown()
6       {
7           OnClick?.Invoke(this.transform.position);
8       }
9 }
```

2.2.5 TileScript

Manages the properties of each tile, including its type, neighbors, and interactions with the active player. It updates dynamically based on gameplay actions like player movement and tile discovery.

```
1 public class TileScript : MonoBehaviour
2 {
3       public TileType TT;
4       public TileScriptableObject TSO;
5       private void AddNeighbours() { ... }
6 }
```

2.2.6 TileScriptableObject

Defines a tile's dynamic properties, such as ownership, whether it is passable, rootable, or contains specific objects. It also defines how the tile is displayed and interacted with.

```
1 public class TileScriptableObject : OnHoverSC
2 {
3       public int id;
```

```
4       public bool passable;
5       public bool rootable;
6       public Vector3 coordinates;
7       public Vector2 ijCoordinates;
8       public Player owner;
9       public GameObject representation;
10      public TileTypesEnum tileTypes;
11      public GameObjectTypeEnum childType = GameObjectTypeEnum.None;
12
13      public override void AskForDetails() { ... }
14  }
```

2.3 Camera and Movement Systems

2.3.1 StrategyCameraControl

Controls the camera's movement, zoom, and rotation. It allows players to pan the camera, zoom in or out, and rotate around specific objects (e.g., players) in the game world.

```
1 public class StrategyCameraControl : MonoBehaviour
2 {
3       public void HandleMovement() { ... }
4       public void HandleScrollZoom() { ... }
5 }
```

2.3.2 Movement

A base class for handling movement across the game world. Players and AI can use this class to move to new positions in the environment. Movement is handled via coroutines to ensure smooth transitions.

```
1 public abstract class Movement : MonoBehaviour
2 {
3       public void Move(Vector3 destination) { ... }
4       protected IEnumerator MoveCoroutine(Vector3 destination) { ... }
5 }
```

3 Tables

3.1 Players and Their Resources

Player	Water	Energy	Biomass	Protein
Player 1	100	50	75	30
Player 2	80	60	90	40
Player 3	120	70	110	50

Table 1: Player resources in "Eva: The Beginning"

3.2 Tile Types and Objects

Tile Type	Object Type	Interactions
Rock	None	No interaction with the player
Grass	Tree	Can be planted
Sand	Bush	Obstacle to pass
Water	None	Can be used for irrigation

Table 2: Tile types and objects in "Eva: The Beginning"

3.3 Game Settings

Setting	Value
Map Size	6x6
Cycles per Day	2
Difficulty Level	1 (Easy)
Rooting Only Next to Self	Yes

Table 3: Game settings in "Eva: The Beginning"

4 Last Chapter

4.1 Conclusions

The game architecture of "Eva: The Beginning" provides a flexible and scalable framework for game development. By integrating Unity with C#, it ensures efficient handling of player actions, world-building, and resource management. The modular design allows easy expansion and modification of game features, making it ideal for iterative development and feature addition. [1–6]

In terms of game design, principles from books like Jesse Schell's *The Art of Game Design* [5] are applied to ensure engaging mechanics that encourage strategic thinking. Furthermore, applying design patterns [1] enhances the maintainability and scalability of the game's codebase.

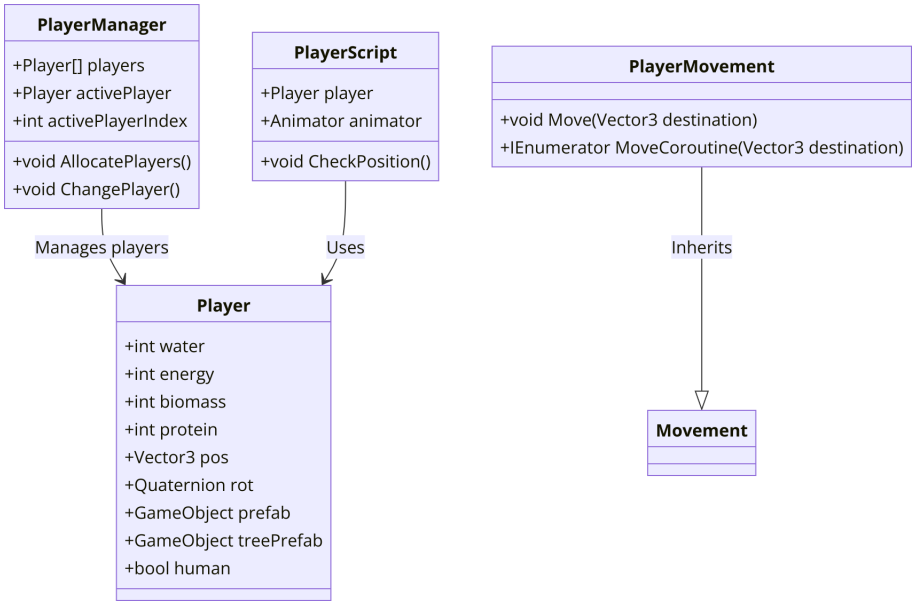

Figure 1: UML diagram for Player Management

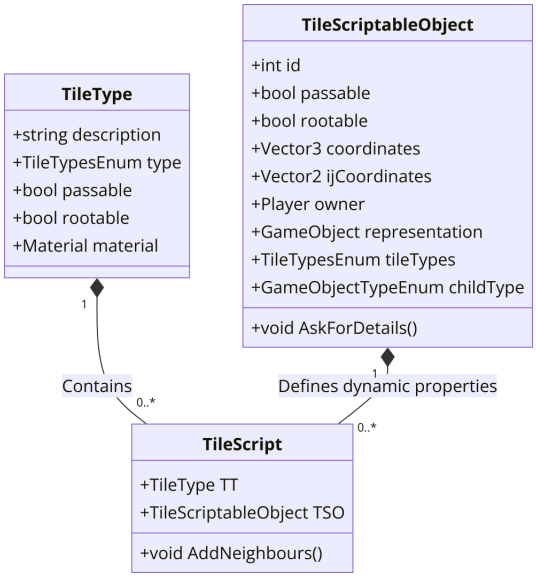

Figure 2: UML diagram for Tile Management

References

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[2] M. Geig. *Unity Game Development Cookbook: Essentials for Every Game*. Packt Publishing, 2017.

[3] Microsoft. *C# Documentation*, 2024. Accessed: 2025-01-04.

[4] I. Millington. *Game Programming: The Next Level*. Apress, 2nd edition, 2009.

[5] J. Schell. *The Art of Game Design: A Book of Lenses*. CRC Press, 2nd edition, 2014.

[6] U. Technologies. *Unity User Manual*, 2024. Accessed: 2025-01-04.