B2B Brera Museum App - Technical Design Document

Release link

https://github.com/alien089/B2B-Museum/releases/tag/v.1.0.2

Section 1 - Introduction

This is the Technical Design Document for the project of an application of gamification of an art museum. Inside this document you can find the features, the diagrams and the logic of this game.

Section 2 - Features

- Player
    - Movement by anchored points
    - Interactions with pictures
    - Camera rotation and zoom
- Puzzles
    - Dragging finger to draw a line for explore the puzzle
    - Multiple start position
    - Object of the puzzle to collect
    - Multiple puzzles to close a level
- Augmented Reality
    - Scan of picture in real life

Section 3 - Choice of Game Engine

The game engine that has been chosen is Unity.
This engine posses different interesting features like:

- A useful mobile touch input system
- An intuitive editor
- Data Management
- interesting plug-in's

The reason for that choice is the target device of the application, the mobile devices. On these devices other engines would be unsuitable. We had chosen the 2022.3.10f version because this is the standard version adopted by the company.

Section 4 - External Packages

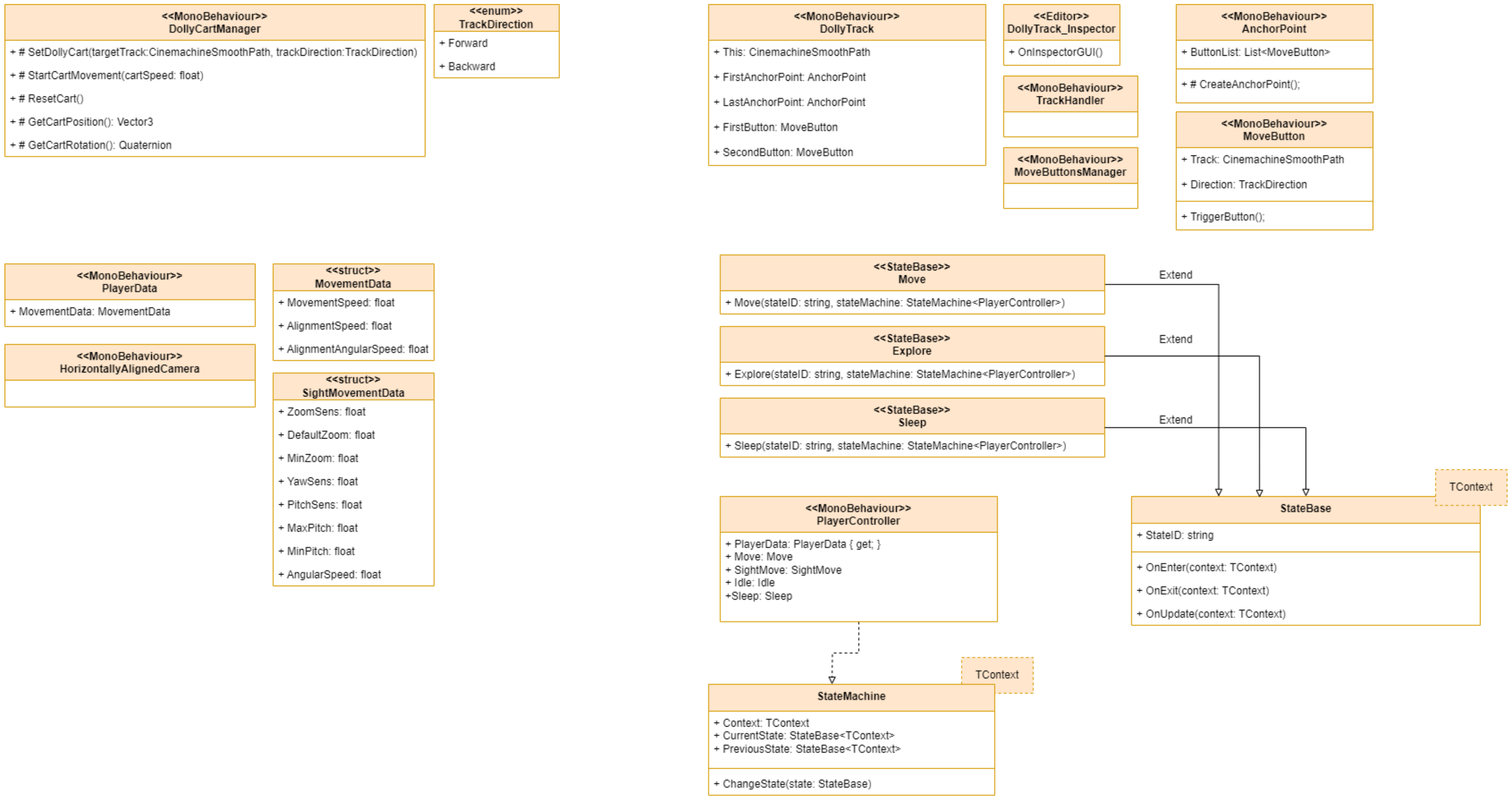
4.1 Vuforia Engine

For this project we have needed of an AR SDK, so we choose for Vuforia Engine, an SDK implemented for provide augmented reality functionality starting from the use of the device's Camera, to the storage of images and 3D models to be scanned into a database so as not to weigh on the project. In addition to this, Vuforia Engine independently generates its part within the application manifest to then request the user to access the camera

Section 5 - Diagrams

5.1 Player Diagram

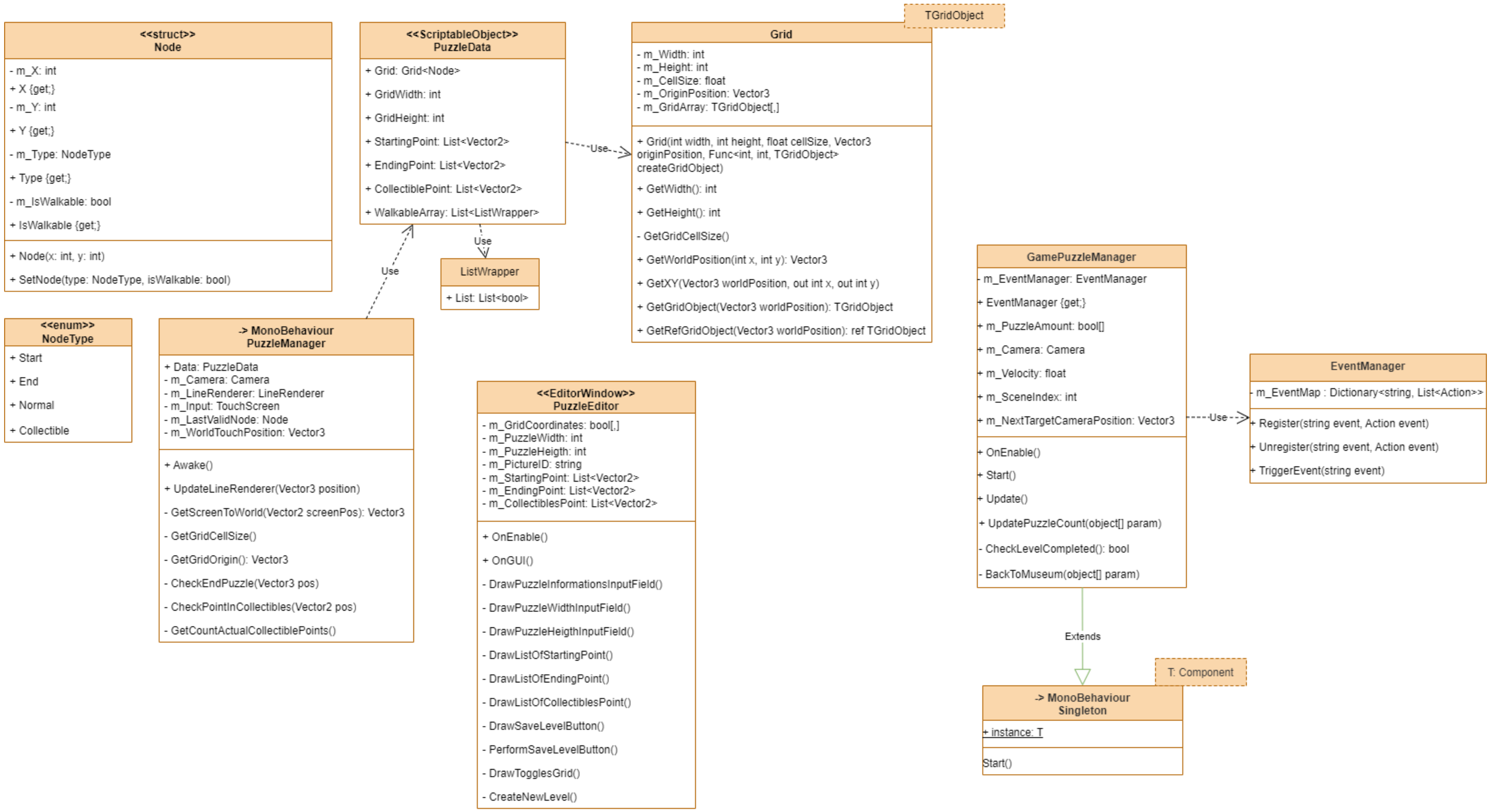
Player Diagram



<<MonoBehaviour>>
DollyCartManager
+ # SetDollyCart(targetTrack:CinemachineSmoothPath, trackDirection:TrackDirection)
+ # StartCartMovement(cartSpeed: float)
+ # ResetCart()
+ # GetCartPosition(): Vector3
+ # GetCartRotation(): Quaternion
<<enum>>
TrackDirection
+ Forward
+ Backward
<<MonoBehaviour>>
DollyTrack
+ This: CinemachineSmoothPath
+ FirstAnchorPoint: AnchorPoint
+ LastAnchorPoint: AnchorPoint
+ FirstButton: MoveButton
+ SecondButton: MoveButton
<<Editor>>
DollyTrack_Inspector
+ OnInspectorGUI()
<<MonoBehaviour>>
TrackHandler
<<MonoBehaviour>>
MoveButtonsManager
<<MonoBehaviour>>
AnchorPoint
+ ButtonList: List<MoveButton>
+ # CreateAnchorPoint();
<<MonoBehaviour>>
MoveButton
+ Track: CinemachineSmoothPath
+ Direction: TrackDirection
+ TriggerButton();
<<MonoBehaviour>>
PlayerData
+ MovementData: MovementData
<<MonoBehaviour>>
HorizontallyAlignedCamera
<<struct>>
MovementData
+ MovementSpeed: float
+ AlignmentSpeed: float
+ AlignmentAngularSpeed: float
<<struct>>
SightMovementData
+ ZoomSens: float
+ DefaultZoom: float
+ MinZoom: float
+ YawSens: float
+ PitchSens: float
+ MaxPitch: float
+ MinPitch: float
+ AngularSpeed: float
<<StateBase>>
Move
+ Move(stateID: string, stateMachine: StateMachine<PlayerController>)
<<StateBase>>
Explore
+ Explore(stateID: string, stateMachine: StateMachine<PlayerController>)
<<StateBase>>
Sleep
+ Sleep(stateID: string, stateMachine: StateMachine<PlayerController>)
Extend
Extend
Extend
TContext
StateBase
+ StateID: string
+ OnEnter(context: TContext)
+ OnExit(context: TContext)
+ OnUpdate(context: TContext)
<<MonoBehaviour>>
PlayerController
+ PlayerData: PlayerData { get; }
+ Move: Move
+ SightMove: SightMove
+ Idle: Idle
+Sleep: Sleep
TContext
StateMachine
+ Context: TContext
+ CurrentState: StateBase<TContext>
+ PreviousState: StateBase<TContext>
+ ChangeState(state: StateBase)

5.2 Puzzles Diagram

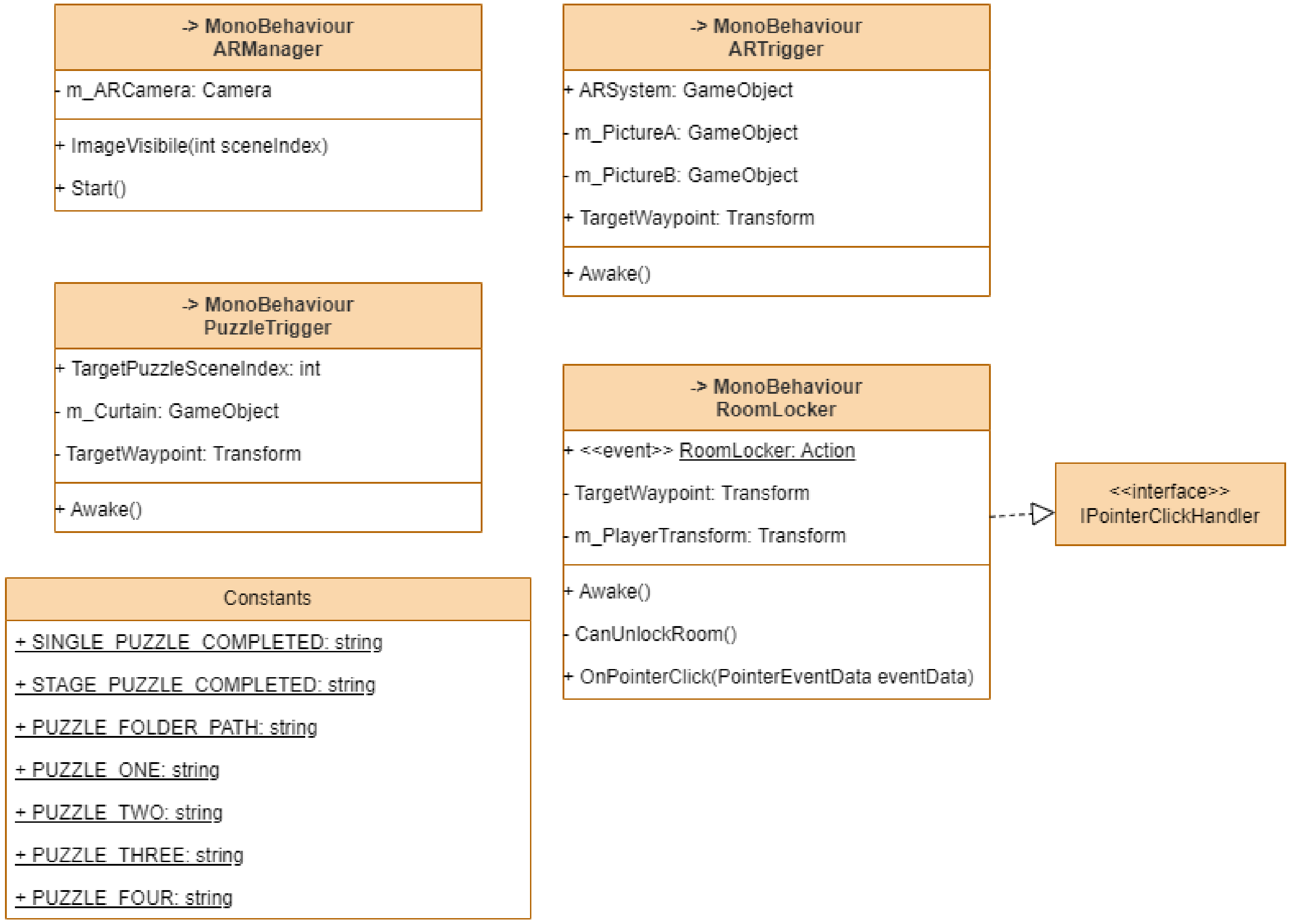
UML B2B PUZZLE DIAGRAM



<<struct>>
Node
- m_X: int
+ X {get;}
- m_Y: int
+ Y {get;}
- m_Type: NodeType
+ Type {get;}
- m_IsWalkable: bool
+ IsWalkable {get;}
+ Node(x: int, y: int)
+ SetNode(type: NodeType, isWalkable: bool)
<<ScriptableObject>>
PuzzleData
+ Grid: Grid<Node>
+ GridWidth: int
+ GridHeight: int
+ StartingPoint: List<Vector2>
+ EndingPoint: List<Vector2>
+ CollectiblePoint: List<Vector2>
+ WalkableArray: List<ListWrapper>
Use
TGridObject
Grid
- m_Width: int
- m_Height: int
- m_CellSize: float
- m_OriginPosition: Vector3
- m_GridArray: TGridObject[,]
+ Grid(int width, int height, float cellSize, Vector3 originPosition, Func<int, int, TGridObject> createGridObject)
+ GetWidth(): int
+ GetHeight(): int
- GetGridCellSize()
+ GetWorldPosition(int x, int y): Vector3
+ GetXY(Vector3 worldPosition, out int x, out int y)
+ GetGridObject(Vector3 worldPosition): TGridObject
+ GetRefGridObject(Vector3 worldPosition): ref TGridObject
Use
ListWrapper
+ List: List<bool>
Use
<<enum>>
NodeType
+ Start
+ End
+ Normal
+ Collectible
-> MonoBehaviour
PuzzleManager
+ Data: PuzzleData
- m_Camera: Camera
- m_LineRenderer: LineRenderer
- m_Input: TouchScreen
- m_LastValidNode: Node
- m_WorldTouchPosition: Vector3
+ Awake()
+ UpdateLineRenderer(Vector3 position)
- GetScreenToWorld(Vector2 screenPos): Vector3
- GetGridCellSize()
- GetGridOrigin(): Vector3
- CheckEndPuzzle(Vector3 pos)
- CheckPointInCollectibles(Vector2 pos)
- GetCountActualCollectiblePoints()
<<EditorWindow>>
PuzzleEditor
- m_GridCoordinates: bool[,]
- m_PuzzleWidth: int
- m_PuzzleHeigth: int
- m_PictureID: string
- m_StartingPoint: List<Vector2>
- m_EndingPoint: List<Vector2>
- m_CollectiblesPoint: List<Vector2>
+ OnEnable()
+ OnGUI()
- DrawPuzzleInformationsInputField()
- DrawPuzzleWidthInputField()
- DrawPuzzleHeigthInputField()
- DrawListOfStartingPoint()
- DrawListOfEndingPoint()
- DrawListOfCollectiblesPoint()
- DrawSaveLevelButton()
- PerformSaveLevelButton()
- DrawTogglesGrid()
- CreateNewLevel()
GamePuzzleManager
- m_EventManager: EventManager
+ EventManager {get;}
+ m_PuzzleAmount: bool[]
+ m_Camera: Camera
+ m_Velocity: float
+ m_SceneIndex: int
+ m_NextTargetCameraPosition: Vector3
+ OnEnable()
+ Start()
+ Update()
+ UpdatePuzzleCount(object[] param)
- CheckLevelCompleted(): bool
- BackToMuseum(object[] param)
Use
EventManager
- m_EventMap : Dictionary<string, List<Action>>
+ Register(string event, Action event)
+ Unregister(string event, Action event)
+ TriggerEvent(string event)
Extends
T: Component
-> MonoBehaviour
Singleton
+ instance: T
Start()

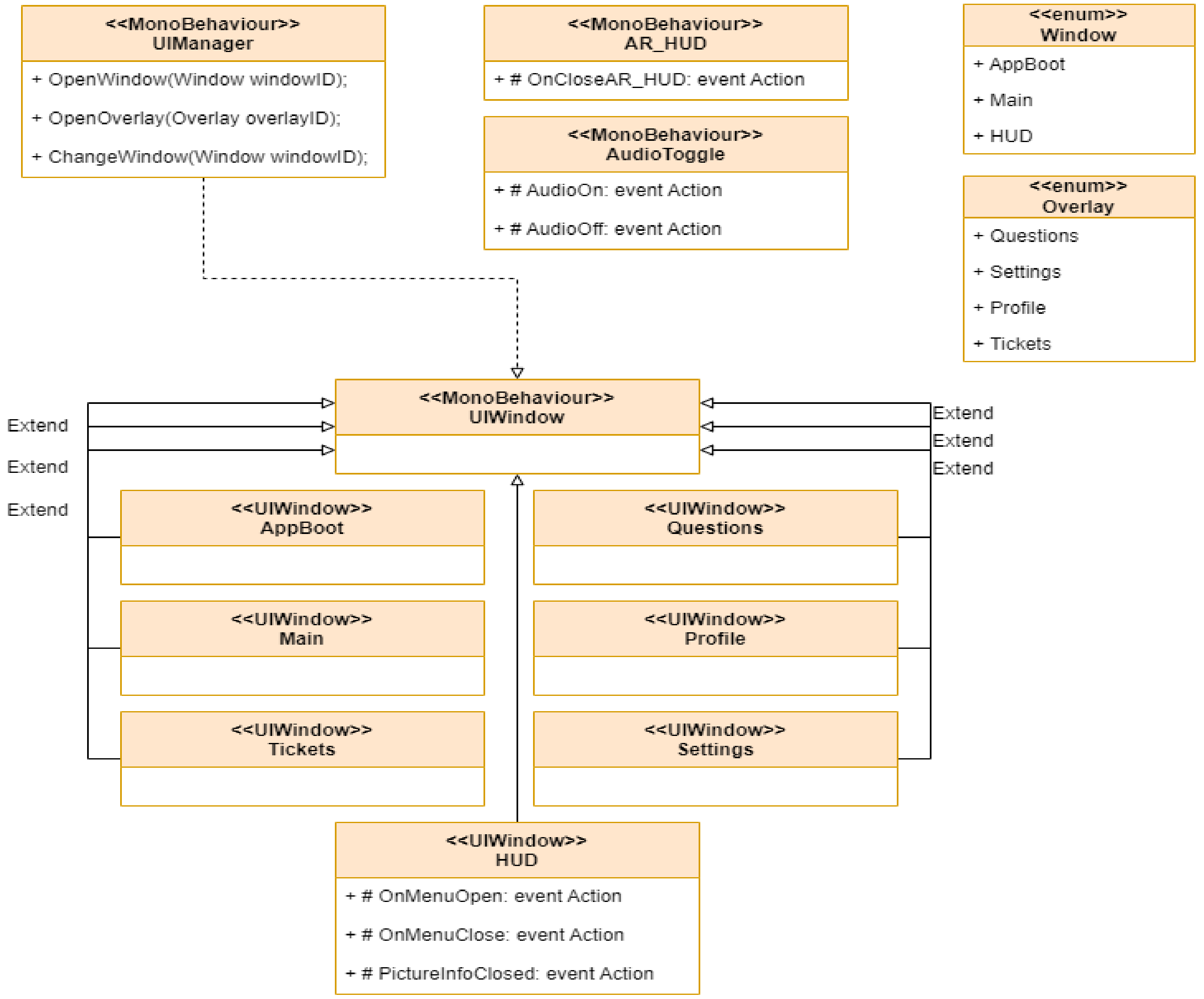
5.3 Misc Diagram

UML B2B MISC DIAGRAM



5.4 UI Diagram

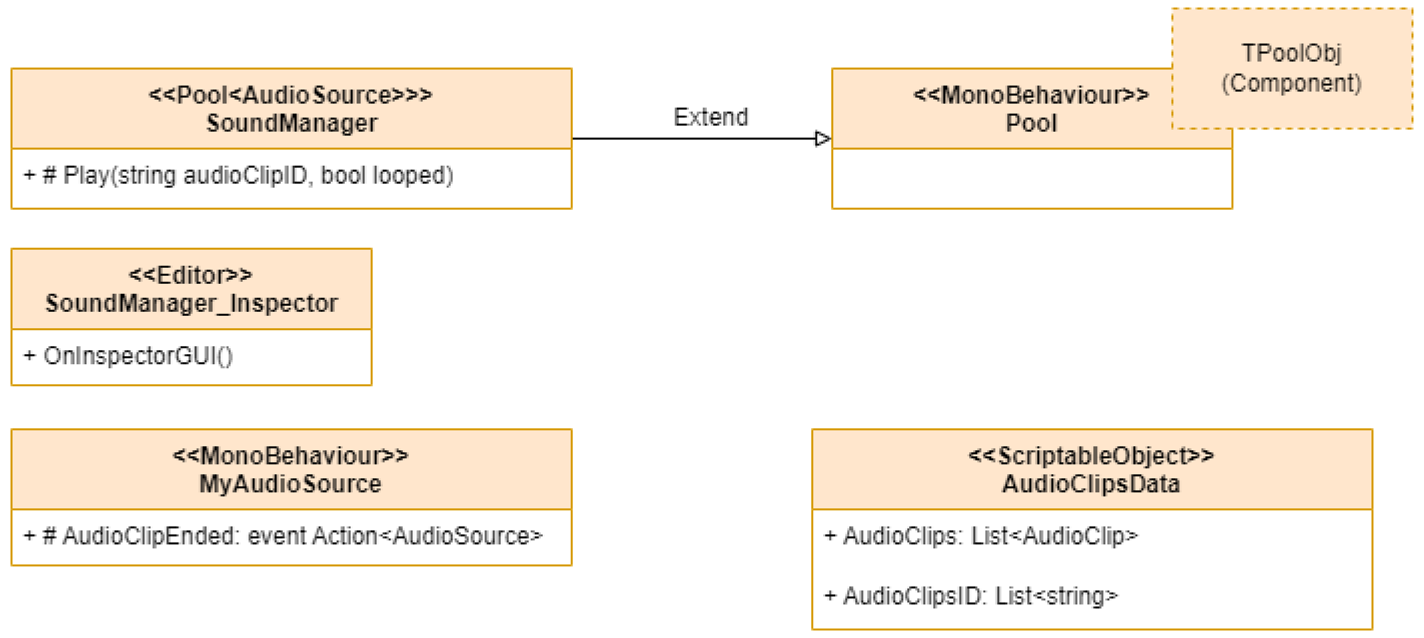
UI DIAGRAM



<<MonoBehaviour>> UIManager
+ OpenWindow(Window windowID);
+ OpenOverlay(Overlay overlayID);
+ ChangeWindow(Window windowID);
<<MonoBehaviour>> AR_HUD
+ # OnCloseAR_HUD: event Action
<<MonoBehaviour>> AudioToggle
+ # AudioOn: event Action
+ # AudioOff: event Action
<<enum>> Window
+ AppBoot
+ Main
+ HUD
<<enum>> Overlay
+ Questions
+ Settings
+ Profile
+ Tickets
<<MonoBehaviour>> UIWindow
Extend
Extend
Extend
Extend
Extend
Extend
<<UIWindow>> AppBoot
<<UIWindow>> Questions
<<UIWindow>> Main
<<UIWindow>> Profile
<<UIWindow>> Tickets
<<UIWindow>> Settings
<<UIWindow>> HUD
+ # OnMenuOpen: event Action
+ # OnMenuClose: event Action
+ # PictureInfoClosed: event Action

5.5 Sound Diagram

Sound System DIAGRAM



Section 6 - Input

For this project we pondered different hypotheses of choices for the input system, after several analyzes we decided to opt for a system that included different typologies:

- [input interfaces] to allow clicking on game objects
- [old input system] to handle situations where constant use was necessary
- [new input system] for use in cases where instant use is necessary

Section 7 - Game Logic

The majority of communication between various classes occurs through "static event" using an approach inspired by the observer pattern, aiming to limit the passing of references between different classes or singletons as much as possible.

7.1 Player

- **Player Controller**:
    - This class manages the state of the player and all the actions that the player can perform during the gameplay:
        - Run the state machine of the player;
        - Memorizes the last position and rotation of the player between scene change thanks to static variables;
- **Player State Machine**:
    - The Player State Machine has 3 states Move, Explore and Sleep:
        - **Move** is the states in which is performed the player movement between one anchor point and another;
        - **Explore** is the state of the player when standing on an anchor point, it managing the rotation and zoom of view, and looks for interactions, with the triggers around the map, performed by the player;
        - **Sleep** is used for disabling all type of interaction with the 3D world by the player

7.2 Movement

- **Movement System**:
    - For the movement, it was decided to use the Cinemachine dolly track and dolly cart because they precisely match the envisioned movement in the game. Therefore, the player only needs to follow the dolly cart to move correctly from one anchor point to another. The movement is performed as follows:
        - Clicking the movement button communicates to the DollyCartManager the target DollyTrack and the target direction along which to execute the movement. Subsequently, the player is transitioned into the "Move" state;

        - In the "Move" state, the player is initially aligned with the transform of the DollyCart, which has been positioned and rotated correctly by the DollyCartManager using data provided by the MoveButton;
        - Once the player has been aligned, the speed of the DollyCart is set, and in the update, the position and rotation of the player are updated to match the position and rotation of the DollyCart until the DollyCart reaches the end of the DollyTrack;
        - Once the movement is completed, the player is returned to the Explore state.
    - The classes that allow for the described functionality are as follows:
        - **DollyCartManager** class is responsible for the managing of the dolly cart that moves on the dolly tracks around the map;
        - **MoveButton** class, attached to the move buttons icons around the map, is responsible for trigger the movement of the player (see in the Explore state class the CheckForWorldInteraction() method);
        - **MoveButtonManager** class is responsible for enabling and disabling the move buttons based on player position;
        - **AnchorPoint** class represents the anchor point where player can explore the area moving the view. Each AnchorPoint contains a list of MoveButtons that are the ones the player can see when located on that specific anchor point;
        - **DollyTrack** / **DollyTrack_Inspector** allows designers to simply position the dolly tracks on the map and connect them to anchor points. All management of the references to MoveButtons is handled automatically, significantly speeding up the level design process.

7.3 UI

- **UI System**:
    - All UI interfaces are classes that inherit from UIWindow, a MonoBehaviour class containing functionalities common to all interfaces. The UIWindows are managed by the UIManager, which holds references to all the windows and categorizes them into two types: Window and Overlay. The difference between a window and an overlay is that an overlay can be opened from multiple points within the entire UI, while a window has a single path to be reached.

7.4 Sound

- **Sound System**:
    - The sounds are managed in the following way:
        - Every time a sound needs to be played, the sound manager retrieves an audio source from the pool and assigns it the requested audio clip.
    - The pool pattern is used for the following reasons:
        - To avoid checking if an audio source is already playing a sound before playing a new one (a new source is simply taken from the pool);
        - The number of audio sources in the scene is variable depending on the demand.

    - All that needs to be done to use this sound system is:
        - pass the references of the audio clips with their corresponding IDs (strings) in the custom inspector of the sound manager;
        - Call the static method SoundManager.Play(string audioClipID, bool looped = false) wherever you want to play the sound.

7.5 Puzzles

- Puzzle Manager:
    - The Manager that manage the correct work of the puzzle, this component allows:
        - the acquisition of data from the PuzzleData
        - The dragging of the user's finger to add to the list of positions of a line renderer, this logic is based on the unity's new input system. The dragging checks also if the requested position is walkable or not
        - Check if in the last position of the line renderer's list positions there is a end position of the puzzle
        - For the Collectible Puzzle check if inside the line renderer's list positions there are all collectible positions of the puzzle
- Game Puzzle Manager:
    - The Manager that controls, when a puzzle manager communicates that it is finished, if all puzzles are completed, and then changes scene
- PuzzleData ScriptableObject:
    - This asset contains the information about a single puzzle:
        - An integer of the Grid Width
        - An integer of the Grid Height
        - A list of Vector 2 to storage the starting cells of the puzzle
        - A list of Vector 2 to storage the ending cells of the puzzle
        - A list of Vector 2 to storage the collectible cells of the puzzle
        - A list of of list to storage the walkable or not walkable cells of the puzzle
- Puzzle Editor:
    - An editor tool for generation of PuzzleData (see its dedicated document)

7.6 Augmented Reality

- VuforiaConfiguration ScriptableObject:
    - This asset contains all the information about the package, the parts that we have used are:

        - the part about the license key
        - the part about the chosen device for the real camera
- AR Camera:
    - The Camera of Vuforia Engine, it includes a link to the VuforiaConfiguration asset and manage the Camera in real life with the continuous creation of a material based on the user’s camera
- AR Image Target:
    - The Vuforia component that manages the acquisition of the linked image. This component permits different uses:
        - Acquisition of the image from folder
        - Acquisition of the image from the online database of the user
        - Acquisition of the image from the Cloud Reco

      This component contains an event system too, it permit to control what happens when the target is found, and what when the target is lost, in our case we used only the first with the AR Manager
- AR Manager:
    - The Manager that controls the scene change when the AR image target recognizes the image in real life

Section 8 - Art Tools

8.1 2D Art

8.1.1 Adobe Photoshop

- Versions: CC 2023
- Field of using:
  - Texture
  - Walls
  - Interface
- About: Graphics Editing Program
- Reason of using:
  - Standard in the industry because its versatility in using by other tools
  - Supporting by the productor
  - Interesting plug-in’s

8.1.2 Procreate

- Versions: 5.3.3
- Field of using:
  - Texturing
  - Coloring
- About: Graphics Editing Program
- Reason of using:
  - Always at hand
  - Easy to use

8.1.3 Adobe Illustrator

- Versions: CC 2023
- Field of using:
  - UI Design
  - Graphic Layout
- About: Graphics Editing Program
- Reason of using:
  - Vectorial drawing
  - Easy integration between images and writing

8.2 3D Art

8.2.1 Maya

- Versions: 2023
- Field of using:
    - Hard surface modeling
- About: Graphics Editing Program
- Reason of using:
    - It’s an extremely powerful 3D modeling software with a lot of professional tools, especially for animating. It’s often used in conjunction with other softwares such as Substance Painter or Zbrush

8.2.2 Adobe Substance 3d Painter

- Versions: 2023
- Field of using:
    - Texturing
    - Baking
- About: Graphics Editing Program
- Reason of using:
    - While still being very user friendly, it allows you to rapidly create high quality textures for a 3D asset.
    - There is a large number of premade materials that speed up the workflow

Section 9 - Conventions

9.1 Coding Naming Convention

Element Type	Convention Choice
Public Class Variable	PascalCase
Private Class Variable	m_PascalCase
Protected Class Variable	m_PascalCase
Local Variabile	camelCase
Property	PascalCase
Constant	SNAKE_CASE
Method	PascalCase

9.2 Assets Naming Convention

[*Prefix*]_[*NameAssets*]_[*Number*]_[*variant*].[*File type*]		
Prefix	Type	AssetName
Doc	Document	GDD(docx)
Doc	Document	LDD
Doc	Document	ADD
Doc	Document	TDD
Doc	Pitch	Pitch
Doc	Document	FC (FlowChart)
Uml	Document	UML (Universal Modelling Language)
Sk	2D Art	Sketch for Concept
C2D	2D Art	Concept for 2D Art
C3D	2D Art	Concept for 3D Model
Sp	2D Art	Sprite
SpSh	2D Art	SpriteSheet
UI	2D Art	UI Sprite
T	2D Art	Texture
An	Anim	Animation
AnC	Anim	Animation Controller
3D	3D Art	3D Model
M	3D Art	Material
Mesh	3D Art	Mesh
Sd	3D Art	Shader
SdG	3D Art	Shader Graph
VFX	3D Art	Visual Effect
Vol	3D Art	Volume
BG	Audio	Background music
SFX	Audio	Sound Effect
PM	3D Art	Physic Material

9.3 Directories Convention

Name	**Usage**
Scenes	This folder contains the scenes. An “Example” folder is used for containing the example scenes.
Animators	This folder contains the animator controllers.
Art	This folder contains the 2D and 3D arts.
Art/2D	This folder contains the 2D art.
Art/2D/Sprites	This folder contains the 2D generic sprites.
Art/2D/Sprites/Puzzles	This folder contains the 2D Puzzles sprites.
Art/2D/Sprites/Curtains	This folder contains the 2D Curtains sprites.
Art/2D/Textures	This folder contains the 2D textures.
Art/2D/Textures/AR Texture	This folder contains the AR recognized textures.
Art/2D/Textures/Original Museum Artworks	This folder contains the 2D pictures images.
Art/2D/UI	This folder contains the UI sprites.
Art/3D	This folder contains the 3D models.
Audio	This folder contains clip audios and mixers.
Audio/Clips	This folder contains clip audios.
Materials	This folder contains materials.
Prefabs	This folder contains prefabs.
Scripts	This folder contains scripts.

Section 10 - Contact on repositories

10.1 Game Design:

- Riccardo Volpi - Smartis93
- Lorenzo Porfirione - JustPorfix

10.2 Game Programming:

- Gabriel Moratelli - Alien089
- Gabriele Garofalo - gabevlogd

10.3 Game Concept Art:

- Martina Paesani - martinapaesani
- Victoria Bedont - Vioosa
- Zhou Xunhan - Uranian3
- Federico Agosti - RuggyApple
- Eva Cividini - evabbe
- Giulia Timeo - giuliatimeo
- Wendy Harrabi - wendy0harrabi
- Romi Rossetti - Scarabaeus97

10.4 Game 3D Art:

- Veronica Passalacqua - VeronicaPassalacqua
- Francesco Rindone - Franceeh