Sorbonne Université
Master ANDROIDE

Développment d'un jeu serieux du projet HUMANI

Stage de Master 1

Réalisé par :

Qingyuan Yao

Encadré par :
Thibault Carron, LIP6, Sorbonne Université

Référent :
Thibault Lust, LIP6, Sorbonne Université

September 15, 2022

Contents

Chapter 1

Introduction

This internship consists of the development of a serious game that raises awareness of the living condition of chamois in the Bauges as well as the impact of human activities (hunting and hiking) on the local fauna. The game is set out to show the research result of the project HUMANI at the University of Savoie.

The game has already been worked on by students in of L3 in 2019. The goal of this internship is to correct any bugs, optimize the execution of the game, add new gameplay elements and create a proper game design document.

The game is developed in C# with Unity.

While the internship is for a project of University of Savoie, the place where I did the internship is at the laboratory of LIP6.

Chapter 2

State of the Art

2.1 Project HUMANI

This game is an attempt to deliver the recent research result from the project HUMANI, "an interdisciplinary project that focuses on how outdoor recreation affect mountain ecosystems and from there on, provide managers with tools to better council the development of outdoor activities with conservation and management of wildlife." [1]. The project divides mainly into 3 axes[2]:

- Sociological and psychological aspects of human activities in the wild
- Behavioral adaptation of the animals according to human activities
- Management of the environment

The game mainly focuses on representing the first two axes, showing the player how hunters and hikers act while in the Bauges and how their actions affect the ecosystem, as well as how the fauna reacts to said activities and how they cohabit with humans.

2.1.1 Humans

In the human aspect, surveys and interviews haven't been carried out in order to analyze visitors of the Bauges in sociological, as well as psychological lenses.

The group of Perrin-Malterre, Chanteloup and Gruas[3] carried out a survey to obtain quantitative data in order to analyze the demographics of the visitors, and searched for links between activities and their social origins. They have also done interviews with 35 persons for qualitative data in order to categorized visitors into 3 types of profiles: Adventurer, Hiker, and Hedonist.

With the previous data, Chanteloup and Perrin-Malterre joined with Marpot[4] to analyze further the mentality of hikers and skiers in the Alps. Observing how the topology affects perceptions, experiences, and choice making of the practitioners of sports.

2.1.2 Chamois

The group of Courbin, Garel, Marchand, Duparc, Debeffe, Börger and Loison[5] set up research camps in the Bauges and put trackers on chamois in order to track their activity

zones and migration activities. They found out that chamois herd move to different places as the season, time of day, and human activities change (more people during the day, hiking in summer, skiing in winter, etc.).

2.1.3 Extra Research

More research has been done by me, as well as the previous developers.

The previous group had the chance of meeting the scientists from project HUMANI, which provided some valuable data on the chamois, as well as regulations in the Bauges (the record can be found in the Docs folder in the git repository[6])

I have also done research concerning the activities proposed in the Bauges, practices of hunting, the current imbalance of preys and predators[7], the hiking routes[8], etc. Some of the information will be used in the game design (cf. A).

2.2 Game Development

Despite having one person in this phase of development, the Scrum method [9] has been implemented to track progress every week.

The game engine is Unity, several technologies have been used to assure the performance and functionalities, notably:

- A* pathfinding [10]
- Asynchronous executions with Unity Coroutine[11] and C# async[12]
- multi-threading with Unity jobs system and DOTS[13]
- Singleton design pattern[14]

Chapter 3

Contribution

In this chapter I'll mainly explain what I've done to enhance the game, however, I'll try my best to explain the implementation of the previous dev team.

3.1 Game Design Document

A game design document is a document that describes everything about a game, from gameplay to design. It does not, however, talk about the programming aspect of the game, but nevertheless serves as a guideline for the programmers. What's the annex is my first attempt to write such documents, I have tried my best to describe the game in a systematic way, which helps explain the code. It is **STRONGLY RECOMMENDED** to read the game design document before continuing forward.

3.2 Game Elements Modifications

This section talks mainly about the big modifications I have made in the Unity editor based on the previous version.

3.2.1 Reorganization

Some files and GameObject structures weren't well-organized in my point of view, so I restructured the Game Scene in Unity, as well as the scripts and sprites in folders.

I also deleted some files (notably the colored map and characters) which could still be retrieved in git.

3.2.2 Assets and Animations

Previously, the art assets were added much later in the game, the developers were using substitutes for most of the time and didn't have time to import all assets into the scene. So I picked up where they have left off, imported the sprites and animation of the wolves, the NPCs, etc.

3.2.3 Collisions

Since the map was just a big image, the colliders had to be added manually, the previous colliders weren't accurate enough and weren't usable at all for the sliding slopes (cf. **??**) mechanism. So I had to readjust and recreate all the colliders and group them by areas for better management.

3.2.4 Re-centering

As each object in Unity has the position data and its position is relative to that of its parent. It is very easy to de-center objects when creating and moving objects (the center of an object is far away or the center of its parent is far away), which is what happened in the scene. So I had to recenter the map to (0,0,0) which includes aligning every tile of the map and it's children accordingly.However, I have ignored the scaling and moving of the elements on the map (trees, NPCs, flags, etc.), which leads to losing the original positions of almost all the objects. Fortunately, the map hasn't been populated too much and the positions of the objects weren't definite. I was able to scale and put objects where I think they fit for the game.

3.2.5 Map Color

Initially there were 3 copies of the map in 3 colors for 3 characters, which seems redundant. Since the colors are monochrome (Only shades of one color, no mixing with other color, cf. A.5.1), I've decided to create a black and white map and add tint to the map according to current character. However, it doesn't look the same as before and the white areas always have a tint instead of being pure white. If there's no way to enhance the look, maybe it's best to just use the old maps.

3.3 Used Assets and Modifications

Several asset packages have been used by me and previous students in order to facilitate the development.

Visual Assets: Most assets are made by the artist in the previous dev team, some third party UI assets had also been used, but I can't trace the source. I later added an asset package called "Fantasy Wooden GUI Free".

Draw2DShapes (deprecated): This package was used by previous students to draw colliders for the mountains, but I found it not precise and not easily modifiable, so I just used the original editor without using the 2D shape editor script.

RPGM: The previous dev team has partially used a package called RPG Maker [15] from the Unity Asset Store, originally, it is used for introducing game development to people who don't know how to code. The code offers a well-built structure for dialog management including scheduler and dialog branches, which is the functionality mainly used by the previous dev team. However, some of the implementations are not optimal for this game, so modifications had to be made. (cf. 3.8.5)

Joystick: The previous dev team used an asset package for joystick control.

Pathfinding and Grid System by CodeMonkey: I used the A* pathfinding code example made by CodeMonkey[16], as well as his Grid class (but it was heavily simplified due to the data-oriented programming (cf. 3.11.3))

3.4 Player Movement

The player movement are mostly made by the previous dev team, I only added the control by keyboard and sped up the movement. The movement of characters was made in Joystick_Link.cs which reads horizontal and vertical input from Joystick.cs, it also passes these inputs to the animator in order to show the correct animation according to the moving direction.

(As the pathfinding mechanism performs well for the wolves, a point-and-click control could be interesting to implement.)

3.5 Interaction

3.5.1 Automated

Automated interactions uses colliders as triggers to detect the player (or the player uses triggers to detect objects). When chamois gets close to a herb, the herb disappear and updates the status of the chamois. When the wolves get close enough to a chamois, the chamois gets bitten and starts bleeding. When the chamois gets onto a slope, the rigidbody is constantly being applied a force downwards. When a bullet hit an object, it compares the tag of the collided object and decides the outcome. etc.

3.5.2 Interactables

The objects and NPCs that need a confirmation (a button) to start interaction are called "Interactables".

Previously, the NPCController class manages showing up the RPGM conversation interface and option buttons on the screen. But as I add more objects with which the player can interact, I feel the need to separate interactive objects and interactive NPCs.

I created a parent class of NPCController called InteractableController, which handles showing up the trigger buttons ("Validate", "Talk", "Pick up", etc.) at the right place on the screen (just above the objects) and assigning the correct actions. The NPCContrller however, handles more actions such as constructing and loading conversation trees (convoTree) and starting conversations (calls ShowConversation.cs which launches the VisualNovel interface).

3.6 Pathfinding

The pathfinding has been considered for the wolves, which when on patrol mode, move randomly between their patrols points, since originally their movement was only in direct lines, which collides with objects all the time. (For the moment, chasing the chamois

and fleeing from humans are still straight line movement, these movements could also implement pathfinding)

After looking for pathfinding system that already exists in the Unity asset store, I've found out that they are either paid, only built for 3D environment, doesn't work with the map, or simply too slow to run. The last problem could be explained by the size of the map, which is a 600x600 grid, and it is impossible to vector it. In the end I've decided to take the grid-based A* pathfinding[10] code of CodeMonkey[16], and optimize for the game.

For each agent, the pathfinding system search for a path on the 600x600 grid, which returns an acceptable precision. But when used on all wolves, the game starts to drop frame, even freeze up completely sometimes. This is expected as there are too many processes executing at the same time, and sometimes it's never possible for a wolf to find a path.

I've tried to optimize the pathfinding at maximum by:

- Baking a bitmap of what's walkable and what's not by box-casting on the colliders.
- Reducing the size of the grid to 300x300
- Limiting the path search cycles to 700 iterations max
- Using Unity Job System (cf. 3.11.3) to access multi-threading functionality for the calculation of paths
- Using sync (cf. 3.11.3) for pathfinding, so the system won't have to wait for the result of the calculation.
- Disabling movement of the wolves if they are too far from the player (cf. 3.11.2)

3.7 FogOfWar

The FogOfWar was previously used in the gameplay of the chamois and the hiker for exploration. But as the chamois lives in the mountains, I feel it doesn't make sense for the chamois to discover its own habitat. So the current version only has the Fog Of War for the hiker.

The system was previously built by (presumably) following a YouTube tutorial[17]. This mechanism uses camera culling in the classic render pipeline (which stopped working when I tried to switch to the new render pipeline for creating shader graphs). I left the mechanism as it is for the most part.

What I've done mostly is for the counting of explored percentage. Previously, the system counts every single pixel on the mask which is a black RenderTexture of 2048x2048, which takes a long time to calculate and freezes the game. I optimized this process by:

- Making the calculation asynchronous
- Using one RenderTexture (2048x2048) for looking good on camera, and another one (100x100) for counting (doesn't need the precision of 2048x2048)
- Separating the map into areas. When leaving one area, the game automatically calculates individual percentage and adds all the percentage together.

3.8 UI

There is an UIManager.cs which controls the switching between interfaces.

3.8.1 Home Menu

This is the menu before the game scene, which lets the player choose characters, view credit, toggle options (notably clearing saves), and quit the game.

When a character is selected, Global.cs will record the selected character, which would be used in the game scene, and then a loading screen will appear, showing the progress while the scene being loading asynchronously (SceneManager.LoadSceneAsync()).

3.8.2 Menu

When the menu button is clicked, the game pauses, the game UI (joystick, date) disappears, replaced by a row of character icons and a column of buttons for launching minimap, encyclopedia, achievements&quests, home menu. The pausing is managed by PauseMenu.cs. When paused, the game will automatically save (but this could impact performance, especially when saving FogOfWar (cf. 3.10)).

When the character icons are clicked, the current character in Global.cs will change and the scene will reload to switch to the chosen character.

The home button returns to the home menu.

The minimap button shows the minimap as well as the player's position on the map.

For the encyclopedia button and achievements&quests button, a sub menu will appear.

3.8.3 Encyclopedia

All the entries are read at the start of the game from a json file and stored in dynamicInfo and staticInfo. All the content from staticInfo will be immediately added to pagesStatic which will appear as the manual; for dynamicInfo, only when an event is triggered will the corresponding entry be added to pagesDynamic, which shows up as notes.

The previous dev team has created 3 encyclopedias GameObject for each character along with 3 copies of sub-menu and buttons. I've tried my best to simplify them, but some aspects are still redundant due to the nature of the structure and the code.

3.8.4 Achievements and Quests

For achievements, I didn't modify too much or check if everything works properly, but I did optimize some redundant references (and corrected the "Achievment" typo...).

For quests, I constructed PlayerQuest.cs and QuestManager.cs from the ground up.

The quests are read from a json file by QuestManager and constructed into PlayerQuest objects (Quest.cs was occupied by RPGM), which contains title, description, hashtable of hints, number of steps it takes to complete, name of NPCs needed for the quest, next linked quest, and status showing whether the quest has been completed.

Currently, there are three types of quest:

- killQuest: Need to kill the correct target
- zoneQuest: Need to reach a zone on the map
- randoQuest: Need to validate all the hiking control flags in the correct order

3.8.5 VisualNovel

The original dialog interface was offered by RPGM[15], the dialog window is very small, especially for mobile platform. So I've decided to remake a VisualNovel interface.

The dialog system of the RPGM had been conserved, but the interface manager has been replaced by VisualNovel.cs and VNLayout.cs.

3.8.6 GameOver

The GameOver canvas can be triggered by several conditions:

- Chamois dies
- Huntress kills an NPC
- Hunter uses up all her ammo (to be discussed)
- A character finishes all the quests

The interface will also display some data from the gameplay, such as the accumulated points, high score, surviving time, etc. These data are mainly recorded by DataStorer, which I haven't finished adapting from the old version, so for the moment it's broken.

3.9 Json

In order to create and import dialog faster (the original RPGM only offers dialog creation in the inspector), the previous dev team has created a system that reads json files and import into the RPGM dialog system. But it turns out the import process was not automatic at all. Every line number has been referenced in the code, which is to say the team has imported the json file manually line by line. I did not understand the purpose of this decision, but since following this implementation makes it too redundant to add new dialog (the current conversations require 2000+ lines of code), I've decided to automatize the process by using loops in the code. I've also changed the structure of the json files for fewer lines and more flexible dialog options.

For other types of json, I did similar things.

3.9.1 Automated Hint Actions

Previously, there were hints used for triggering certain event (changing dialog of an NPC for next time, adding an entry in encyclopedia, validating a hiking control point, etc.), but they were all done manually, which means that for every hint created, there needs to be a switch case which calls a function (and the functions were all made manually for each hint).

For simplifying this with the least effort, I made a "command language" which is read in ShowConversation.cs and calls an action depending on the first word and the current character, the corresponding action function will then read the rest of the commands. There are four types of actions: default action for each character, switchNode, startRando and questAction. The interpretation goes like this:

actionChamois/actionChasseur/actionRando

"NameOfTheEncyEntry" (ex. "infoRegVehicule")

This adds the corresponding entry from dynamicInfo to pagesDynamic.

switchNode

"node,NPCName,NodeName" (ex. "node,GuideRando,2")

This simply switches the firstNode in the convoTree of the NPC.

actionRando

"rand+NameOfTheHikingRoute" (ex. "randMorbier")

This starts a hiking route (setting up dialog for control flags, changing firstNode of GuideRando and updating quest), the name is set this way in order to access directly the GameObjects with the same name.

questAction

"quest,NPCYouJustTalkedTo,NPCToUpdateNode,QuestName"
(ex. "quest,DonneurDeQuete,GardeForestier,UnChamoisMalade")

The quest action needs to update the firstNode of the NPC the player just finished talking to, as well as the firstNode of the next NPC the player need to talk to depending on the quest, the last parameter updates the content in QuestManager. It's possible for the second and third parameter to be empty.

3.10 Save and Load

As this is a considerably long game which doesn't reset the player's progress each time it reloads the scene, a save and load system is needed.

The structure of the system has already been built by previous student following the tutorial of Brackeys[18] (or at least I assume) but it didn't seem to be working properly while testing the game. So I used their system, added and adapted data that need to be saved.

- **Positions**: As Vector3 is not serializable, it has to be translated to 3 floats in order to be saved.
- **ForOfWar**: Saving the textures of FogOfWar is the most difficult to do, it contains several processes of translations between types of object. When saving, I first need to convert RenderTexture to Texture2D, then I would need to EncodeToPNG in

order to obtain an array of byte to save in the system. When loading, I basically do the opposite and replacing the RenderTexture in the scene by the newly converted one.

- **NPC Conversations**: As the convoTree will be reloaded from json, I only need to save the firstNode of each NPC.
- **Encyclopedia**: As content of the entires will be reloaded from json, I only need to save pagesDynamic
- **Achievements**: Only need to save if an achievement has been unlocked (a boolean for each)
- **DataStorer**: I made the class of DataStorer serializable by converting dictionaries to hashtable, this allows me to simple store the whole object.
- **Quest**: Only need to save foundQuests, which is a list of PlayerQuest (I made sure the class is serializable)

3.11 Optimizations

3.11.1 GOPointer and Singleton

Originally, the previous dev team has used GameObject.Find for almost all objects, which whenever it's called, compares the name with every single object in the scene and returns it. As there are hundreds of objects in the scene and hundreds of GameObject.Find in the code. This caused the game to load and run extremely slowly and makes it complicated to change object names. So the first optimization I've done, is replacing most of the GameObject.Find with GOPointer, a script I created myself that contains the link to every previous called GameObject in the code, which only needs to be linked manually once upon declaration. This sped up a lot the loading time.

GOPointer.cs

```
public class GOPointer : MonoBehaviour{
    //Inspector
    public GameObject _PlayerChamois;
    ...

    //Static
    public static GameObject PlayerChamois;
    ...

    public async Task LinkAync(){
        PlayerChamois = _PlayerChamois;
        ...
    }
}
```

Later into the C# development, I have learned that for some classes that only have one instance in the scene, it is easier to access by creating a Singleton in the code. This replaced some references in GOPointer.

Map.cs

```
private void Awake(){
    if(Instance == null) Instance = this;
    else Destroy(gameObject);
    ...
}
```

3.11.2 DisableIfFarAway

It is a general practice to disable objects when the camera is far away, it has already been done by previous dev team, I've added more elements to this process such as wolves with pathfinding and colliders.

3.11.3 Asynchronization and Multi-threading

For tasks to run in parallel, we can either use asynchronous execution with built-in Unity Coroutine, and async from C#, or multi-threading with the Unity Job System.

Unity Coroutine

The Coroutine in Unity allows tasks to be executed asynchronously, but not multi-threadedly, it achieves this by (oversimplified explanation) returning a period of time at a certain point of execution (usually at the start or end of a loop), which allows the engine to return to the main execution for the given time and come back to the Coroutine until the next return or the end.

Here's an example of how the VisualNovel shows the text like a typewriter using Coroutine.

VNLayout.cs:

```
private void setDialog(){
    ...
    tmpCoroutine = StartCoroutine("PlayText");
}

IEnumerator PlayText(){
    string story = currentText;
    foreach (char c in story) {
        textMeshPro.text += c;

        //return to main execution for 0.02s and come back
        yield return new WaitForSecondsRealtime(0.02f);
    }
    ...
}
```

C# async

Apart from Coroutine in Unity, the C# async mechanism offers more flexibility such as being able to run on non-main thread, await for tasks, and return values from tasks.

Here's an example of how Init.cs launches an async task and awaits it to finish.

NPCManager.cs

```
public async Task loadConvo() {...} //has to be async Task to be awaited
```

Init.cs

```
async void Start() {
    convo = npcManager.loadConvo(); //store the task in the class
    ...
    if(convo!=null) await convo; //wait until Task loadConvo is done
}
```

Unity Job System

Unity DOTS(Data-Oriented Technology Stack) is a way of programming which differs from Object-Oriented Programming, the Data-Oriented nature grants a fast performance and allows it to multi-thread on multicore processors. The technology is better used with the Entity Component System(ECS), but as this game was built in classic OOP, it would be too hard to switch. Nevertheless, I've managed to implement one of the most CPU-demanding task in DOTS with the help of the tutorial by CodeMonkey[16]. Paired with async, this allows the pathfinding to execute asynchronously on another thread while the script on the main thread await for the results.

Here's the simplified code in the script.

Pathfinding.cs

```
//Unlike Coroutine, async allows execution on non-main threads and can
    return variables
public async Task<List<Vector3>> FindPath(Vector3 startWorldPosition,
    Vector3 endWorldPosition){
    NativeList<Vector3> path = new NativeList<Vector3>(Allocator.TempJob);
        //NativeList is a data-oriented list for better performance
    ...
    FindPathJob findPathJob = new FindPathJob{
        //Pass the parameters like this
        startPosition = new int2(startX, startY),
        endPosition = new int2(endX, endY),
        pathVectors = path,
        ...
    };
    findPathJob.Schedule().Complete(); //schedule and complete the job
        (multiple jobs can be scheduled)
    ...
    path.Dispose(); //NativeList has to be disposed after finishing the job
    if (pathList.Count > 0) return pathList;
    return null;
}

private struct FindPathJob : IJob {

    public int2 startPosition;
```

```
    public int2 endPosition;
    public NativeList<Vector3> pathVectors;
    ...
    public void Execute() {...}
}
```

3.11.4 Initiation Order

Unity has its own execution orders[19], notably Awake() is called only once upon initiation, Start() is called after Awake() and after being enabled, Update() is executed at every frame.

Upon loading of the game, there are many objects initiating at the same time, which can get heavy for the processor and cause some synchronization problems. Also, it seems that when switching to another scene and reload the scene before, will only activate Start() but not Awake().

These problems have been mostly addressed by using Awake() mostly for creating Singletons (cf. 3.11.1) and in Init.cs which manages the main part of initialization. Furthermore, to guarantee an object has already been initialized when needed, the await mechanism in async (cf. 3.11.3) has been used extensively to wait for the object to be ready before executing.

3.11.5 DontDestroyOnLoad

As the map is consistent for all characters throughout the gameplay, there is technically no need to reload it every time we load a scene. Which is why I have decided to use DontDestroyOnLoad in the Awake function, and if there's another object being created it will be destroyed upon initiation and before the rendering, this makes switch between characters faster as it only required loading the map once when the player enters the scene the first time.

(Other consistent objects could also use this optimization)

3.12 Other Minor Fixes

- Date system: replaced the original home-made date counting system by the C# DateTime
- Shorter button action list: A lot of buttons used to have a series of actions added manually in the editor, which could have been just written in code and would be much easier to modify.

Chapter 4

Conclusion

In this internship I have fixed bugs left by the previous dev team, optimized execution of the game and also added several features. It is a shame that I didn't do enough game design as I've originally envisioned, due to the underestimation of the debug process and the complexity of the subject (HUMANI). However, thanks to this internship, I have done much research and self-learned a lot in C# programming as well as game development. Hopefully it would come in handy for the upcoming semester and in my future career.

During the development I have tried many times to rebuild or change drastically what the previous dev team had made, whether it's because they used a package that I didn't understand, or they implemented something in an inflexible and unoptimized way. But the more I tried to fix thing, the more the game breaks, because I didn't see the bigger picture to avoid breaking links between scripts. If I had tried contacting the previous dev team or the members of project HUMANI, maybe thing would have been easier to understand. But maybe understanding and adapting others' code is also a crucial competence to have as a programmer.

If I were to continue the development, I would focus on these aspects:

- Comment and clean the code, untangle and deleted unused scripts
- Save more data, such as player stats
- Fix DataStorer
- The wolves and wounded chamois need smarter behavior
- Add point-and-click control?
- Created hiking routes based on the real world
- Add the chamois baby
- Have a field trip to the Bauges and meet the HUMANI team
- Game design: continue on the gamification inspired by the papers of HUMANI, and start constructing levels and storylines
- The graphics and UI design need serious work, as well as the audio

Bibliography

[1] *ANR (2019 - 2022) : HUMANI*. URL: https://leca.osug.fr/ANR-2022-2019-HUMANI (page 2).

[2] *Cohabitation entre pratiques récréatives et faune sauvage*. URL: https://cohab.sciencesconf.org/resource/page/id/1 (page 2).

[3] Clémence Perrin-Malterre, Laine Chanteloup, and Léna Gruas. "Outdoor recreation in a Regional Park: types of hikers, ski tourers and snowshoers in the Hautes-Bauges (Savoie, France)". en. In: *Annals of Leisure Research* 24.2 (Mar. 2021), pp. 209–227. ISSN: 1174-5398, 2159-6816. DOI: 10.1080/11745398.2019.1682016 (page 2).

[4] Stéphane Marpot, Laine Chanteloup, and Clémence Perrin-Malterre. "Expériences paysagères et pratique du ski de randonnée dans les Alpes françaises". fr. In: *Projets de paysage* 25 (Dec. 2021). ISSN: 1969-6124. DOI: 10.4000/paysage.24604. URL: http://journals.openedition.org/paysage/24604 (page 2).

[5] Nicolas Courbin, Mathieu Garel, Pascal Marchand, Antoine Duparc, Lucie Debeffe, Luca Börger, and Anne Loison. "Interacting lethal and nonlethal human activities shape complex risk tolerance behaviors in a mountain herbivore". en. In: *Ecological Applications* (June 2022). ISSN: 1051-0761, 1939-5582. DOI: 10.1002/eap.2640. URL: https://onlinelibrary.wiley.com/doi/10.1002/eap.2640 (page 2).

[6] MRVN. *MRVNY/Chamois*. C. June 2022. URL: https://github.com/MRVNY/Chamois (page 3).

[7] *Cerfs, chevreuils, sangliers... Trop d'ongulés nuit aux forêts*. en. Sept. 2022. URL: https://www.onf.fr/vivre-la-foret/+/5a4::cerfs-chevreuils-sangliers-trop-dongules-nuit-aux-forets.html (page 3).

[8] *Réserve Nationale de Chasse et de Faune Sauvage*. URL: https://www.parcdesbauges.com/fr/agir/que-fait-le-parc/valoriser-les-patrimoines/reserve-nationale-de-chasse-et-de-faune-sauvage.html#.Ysf9Ay8Rr5g (pages 3, 22).

[9] *Scrum Guide*. URL: https://scrumguides.org/scrum-guide.html (page 3).

[10] Stefan Edelkamp. "Cost-Algebraic Heuristic Search". en. In: (), p. 6 (pages 3, 7).

[11] Technologies Unity. *Coroutines - Unity Manual*. zh-cn. URL: https://docs.unity.cn/cn/2018.3/Manual/Coroutines.html (page 3).

[12] BillWagner. *Asynchronous programming - C#*. en-us. URL: https://docs.microsoft.com/en-us/dotnet/csharp/async (page 3).

[13] Unity Technologies. *DOTS - Unity's new multithreaded Data-Oriented Technology Stack*. en. URL: https://unity.com/dots (page 3).

[14] *Singleton Design Pattern*. en. URL: https://www.javatpoint.com/singleton-design-pattern-in-java (page 3).

[15] *RPG Maker*. URL: https://learn.unity.com/project/creator-kit-rpg (pages 5, 9).

[16] *Pathfinding tutorial*. Jan. 2020. URL: https://www.youtube.com/watch?v=1bO1FdEThnU (pages 6, 7, 13).
[17] *FogOfWar tutorial*. Oct. 2019. URL: https://www.youtube.com/watch?v=MUV9Nr-cIGU (page 7).
[18] *SaveLoad tutorial*. Dec. 2018. URL: https://www.youtube.com/watch?v=XOjd_qU2Ido (page 10).
[19] Unity Technologies. *Unity - Manual: Order of execution for event functions*. en. URL: https://docs.unity3d.com/Manual/ExecutionOrder.html (page 14).
[20] *"La France comptait 580 loups sur son territoire en 2020"*. URL: https://www.onf.fr/chasse/+/b78::la-chasse-un-prerequis-pour-planter-les-forets-de-demain.html (page 19).
[21] *Trophy Hunting*. en. Page Version ID: 1109106327. Sept. 2022. URL: https://en.wikipedia.org/w/index.php?title=Trophy_hunting&oldid=1109106327 (page 20).
[22] *Chasse à la journée dans la réserve nationale de Chasse et de Faune Sauvage des Bauges*. en. Mar. 2022. URL: https://www.onf.fr/chasse/+/cb4::chasse-la-journee-dans-la-reserve-nationale-de-chasse-et-de-faune-sauvage-des-bauges.html (page 20).
[23] MRVN. *Devlog*. C. June 2022. URL: https://github.com/MRVNY/Chamois/blob/daac798e773321417e84628e3c747852126eded7/devlog.md (page 26).
[24] *Check Lists*. en. URL: https://github.com/MRVNY/Chamois/projects/1 (page 26).

Appendix A

Game Design Document

As the game design for the game hasn't fully finished, I would propose some alternatives ideas in parentheses.

A.1 Introduction

This is a serious game which raises awareness on the effect of human behaviors on the fauna of National Park of Savoie

The game is open-world where the player can explore freely, interact with objects and creatures, and do quests.

The story would be mostly linear, as it is very hard to design a non-linear story and an open world at the same time. Some stories need to be unlocked in order to start other stories. Ideally, the story of three characters would start in parallel, but in certain situation, need the information and interaction from other characters in order to continue the story.

(It would be ideal to have different solutions to a problem (ex. The answer to a question can be found by observing or directly asking the scientist). We can also encourage the player to search online (but in a correct way) as doing research online is a valid way of learning and might inspire curiosity.)

A.2 Characters and Gameplay

There are 3 characters in the game: the chamois, the huntress and the hiker

Figure A.1: Three characters

A.2.1 Chamois

The chamois is a female who is pregnant and needs to survive for as long as possible and give birth to a baby.

Gameplay

There are several stats that keeps the chamois alive: health, hunger, stress, exp, which are affected by these actions (can be discussed and changed in the future):

- Walking: +exp, +hunger
- Eat grass: +health, -hunger, -stress
- Being tripped by trash: -health
- Being next to wolves: +stress
- Being bitten by wolves: -health

The chamois needs to eat and avoid danger in order to keep the stats normal and survive until the birth of the baby.

(The exp was used to enlarge the circle of FogOfWar, but it was canceled since there's no reason for a chamois to discover his own habitat)

As chamois herd move to different places according to the season and time of day, the player should also try to stay close to the herd.

Scenario

Through the eyes of the chamois, we can get to know the behavior of chamois and experience how the human activities have changed the life of animals (for better or for worse)

A.2.2 Huntress

The huntress is a character with a pistol

Gameplay

The huntress needs to talk to NPCs in the park in order to accept quests such as hunting hurt chamois, she's also capable of shooting other creature, but will get a penalty if she shoots the wrong ones. The huntress will only be able to hunt during hunting seasons. The huntress also helps keeping the park clean by picking up trash.

The mechanism of the huntress is not about providing the player an enjoyable shooter game like Hotline Miami, as the usage of guns is a very serious topic in real world. (It might even be possible to simplify the shooting process?)

Scenario

As hunting is a very controversial topic, it's important to teach the function of hunters in regions where predators are in low number (in 2020, there were only 580 wolves in total in France[20]), which is to make up for the lack of predators in order to regulate populations

of certain species. This can be a perfect example of how the human intervention actually maintain the stability of the ecosystem.

However, throughout history, even nowadays, there have been countless cases of trophy hunting[21], which is definitely not what this game encourages. Therefore, behaviors like killing wolves or killing more chamois than expected are to be punished.

We can even implement the ideas of the opposing sides in the game in order to have a debate on these two subjects,

Possible story line: At the start of the game, the girl does an internship on hunting in order to get a license for hunting[22]. At the exam, she needs to answer questions about chamois and the woods (she has to find the questioners everywhere in the mountains). Upon passing the exam, she will be able to freely take requests from NPCs in the woods, but only during hunting seasons. When the hunting season is over, she would have to do other tasks such as helping NPCs in the woods or completing her knowledge on the woods.

A.2.3 Hiker

The hiker does hiking routes and take pictures at each check point.

Gameplay

The hiking routes are obtained from a guide somewhere near the spawning points. After getting a route, he needs to find the check points and validate them. He will also take a photo at the check point (photo provided by future trip to the park)

Scenario

The hiker arrives in the game with an introduction from a guide, offering him routes to discover. The rest of the story hasn't been discussed much.

(As mentioned in State of the Art (cf. 2), the practitioners of sports fit into different profiles and are affected by topology, it may be interesting to evaluate the player on the choices they made in the game.)

A.2.4 Between the Three

It is possible for the 3 characters to interact with each other since it is possible to switch between players in the middle of the game. In that case, we need a system that manages the movement of the player and also prepares the dialog when meeting with another character.

A.3 The World

Even though the map is relatively small, we still need to prevent players from getting lost on the map (there won't be fast travel) or get bored, which means the world would be populated with decorations, interactive objects and NPCS.

A.3.1 Map

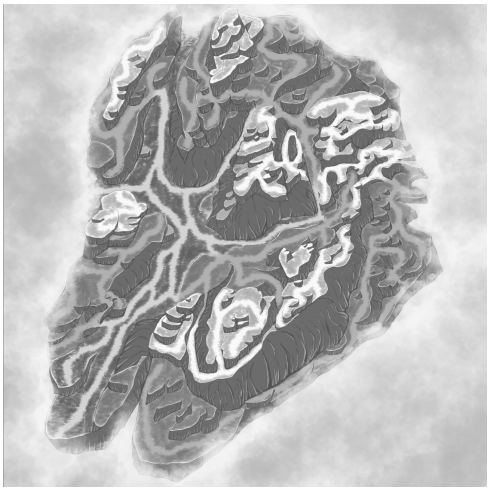

Figure A.2: Map

The map is an adaptation of the real map of the Bauges, which includes slopes and some hills that are only accessible by chamois (ladders or climbing gears could be considered for humans to reach these areas later in the game, to be discussed)

The world will be decorated with trees, bushes, rocks, etc. which are not interactive, but should serve as obstacles or hiding spots.

There are scientists doing research in the park and in these zone, it's only accessible by researchers and the huntress since the huntress hunt for researches, the huntress can get special information (maybe can even be a fast travel point?)

A.3.2 Interactions

There are different types of interactions which the players can give with objects and NPCs.

Objects with Automatic Interactions

- Grass being eaten
- Wolves biting chamois
- Slopes making chamois slide down

Objects with Confirmation-Required Interactions

- Charging ammo
- Trash being picked up
- Trash being thrown into the trash can
- Target being shot by the huntress
- Flag of a control point on the hiking route being validated

NPCs

With human or other chamois, the characters can have a conversation with NPCs in a VisualNovel interface (for readability), here are the current NPCs:

- **InfoGivers**: People who hide in places and give the player a tip when found
- **InfoChamois**: Same as InfoGiver, but for chamois
- **Chamois (The Herd)**: They move to different places according to the season
- **RandoGuide**: Gives the hiker information on hiking routes, should start from simpler routes to more difficult ones. The difficulty of routes can be defined by how complicated to get there (Having a big detour etc. like A Short Hike)
- **WoodKeeper**: As huntress to do errands for him
- **Other hikers**: (Might have prejudice on hunters?)
- **Scientists**: Do research and provide answers

AIs

There is also another type of agent that moves by itself, which are the enemies and wounded chamois (unimplemented). They would both have goals to move towards or away from. The enemies move in an area and hunts down the chamois when it comes, the wounded chamois would need to run away from the huntress.

A.3.3 Other Mechanism

- **Day & Night**: This mechanism has been suspended as there isn't any gameplay that uses this. But it could be envisioned for the variation of location of the chamois herd (cf. 2).
- **Date**: The date passes automatically, but this could be problematic if the player doesn't finish a quest for a long time, and some events require passing to a different season or time of day

A.3.4 Real-World Extensions

The game is envisioned not only to be played, but also to be used in the national park thanks to the tracking by GPS. As players unlock hiking routes which are inspired by real-life routes from a website[8], it would be interesting for people to validate the point in the park using their GPS info.

A.4 Interfaces

A.4.1 Menu

The menu contains buttons which give access to: switching between characters, the achievement/quest menu, the encyclopedia menu, the minimap, as well as returning to title screen. When the menu opens, all activity in the game should suspend for the player to focus on selection.

Figure A.3: Menu

A.4.2 Encyclopedia

The encyclopedia contains "static" and "dynamic", which means, the manual and notes. (for chamois, it could be named differently).

The manual holds all the initial information that the current character needs to know, such as the rules to obey while in the Bauges.

The notes will be updated as the character progress in the game. Several events can add contents to the notes: Talking with an NPC, interacting with an object or triggering an event.

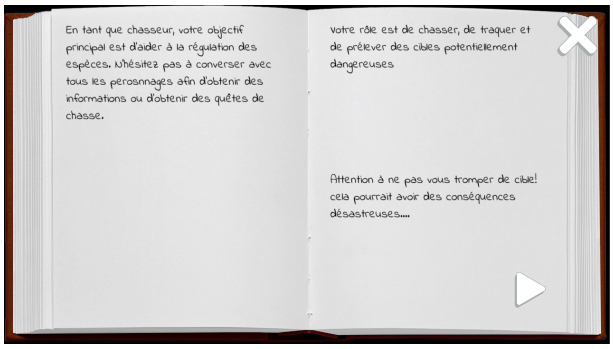

Figure A.4: Encyclopedia

A.4.3 Achievements/Quests

The Achievements and Quests are grouped under the same sub-menu as they both motivate players.

Achievements are permanent goals that track automatically the progress of the play and complete themselves once the requirements are met.

Figure A.5: Achievements

Quests are linked with the story and sometimes has several steps to complete, it also has hints to instruct the player what to do.

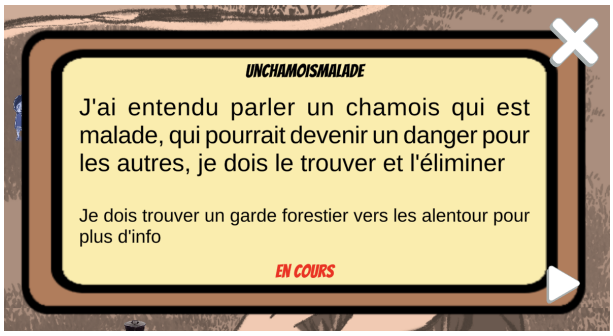

Figure A.6: Quests

A.4.4 Guides

The guide is a window of text that shows up whenever the game needs to inform the player about something. Initially, the guide shows up at the start of the game for each character, but was later replaced by conversations with an NPC in the VisualNovel interface.

A.4.5 VisualNovel

The VisualNovel interface launches whenever a conversation starts. It's the classic style with images of the characters on both sides, a textbox in the lower-center, and options to choose from in the upper-center. The text appears in the style of typewriter.

Figure A.7: VisualNovel

A.5 Style

This part talks about the envisioned style for image, music, writings, etc. As I was not part of the initial dev team and have mostly done the development of this game. My vision on this subject is very limited. I will try to briefly explain how I and the previous students imagined the game to be.

A.5.1 Graphics

In short, the game is in multiple monochromes, which is to say that every object in the scene has only one tint of color that changes according to the scene. This decision was made by the previous developers in order to distinguish the current character (chamois in green, huntress in orange, hiker in blue).

(The game could also change as the season changes (ex. snow in winter, a whiter tint))

The art of the game is mostly drawn by an artist, but some UI assets are used. The style is pretty cartoonish.

The initial UI was a bit messy, I reorganized the status boxes on the screen, and changed some interface images to make it more simplistic. But there are still a lot of room for improvements, especially the fonts.

A.5.2 Audio

The audios are chosen by previous students with a cartoonish style. I cannot elaborate too much as I didn't work on this part of the game.

Appendix B

Scrum

This is a record of the scrum method implemented during the internship, at every start of the week, tasks were set out for the next five days, and at the end of the week, a conclusion of what has been completed and what is left to do.

A devlog[23], as well as checklists[24] are also available on git repository for more detailed progress each day.

End of week 1 (10 June 2022)

Fixes:

- Main Menu: MenuManager
- Easier debug: DebugManager & added keyboard controller
- Optimisation: GameObjectPointer
- UI: little fix of the Achievement UI

Display:

- Sped up walking animation
- Imported wolf animations
- Sorting layer fixed (walk behind trees and bushes)

Collision:

- Separated colliders by zones
- Corrected sliding effect on slopes
- Corrected and added colliders zone by zone

Start of week 2 (13 June 2022)

- Adapt for hunter and hiker
- Explore percentage (Fog)

- Correct interaction with objects
- Enhance UI (pause, fonts, logic, fast character switching)
- Save mechanism
- DialogController replaced by VisualNovel
- Add trees and other props

If time allows:

- start game design
- DisableIfFarAway (done)
- Reduce Awake()

End of week 2 (17 June 2022)

- ☑ Adapt for hunter and hiker
- ☑ Explore percentage (Fog)
- ☐ Correct interaction with objects
- ☑ Enhance UI (pause, fonts, logic, fast character switching)
- ☑ Save mechanism
- ☑ DialogController replaced by VisualNovel
- ☐ Add trees and other props

If time allows:

- ☐ start game design
- ☑ DisableIfFarAway (done)
- ☐ Reduce Awake()

Start of week 3 (20 June 2022)

- Enhancing VisualNovel system
- Adapt interaction button with other objects
- Understand quests and how they're added in encyclopedia
- Save more data
- Path Finding
- Game Design
- Keep fixing collision/sorting layer bugs
- Look into Job System / ECS ?

End of week 3 (24 June 2022)

- ☑ Enhancing VisualNovel system
- ☐ Adapt interaction button with other objects
- ☑ Understand quests and how they're added in encyclopedia
- ☐ Save more data
- ☐ Path Finding (working)
- ☐ Game Design

Extras:

- ☑ Restructure json object
- ☑ Auto import from json
- ☑ Interactables Controller
- ☑ VN for all NPC

Start of week 4 (27 June 2022)

- pathfinding
- Minimap
- Game Design

End of week 4 (08 July 2022)

- ☑ pathfinding
- ☑ Minimap
- ☑ Game Design
- ☑ Generate .apk

Start of week 5 (11 July 2022)

- Game Design
- Construct 1 quest for each perso
- More save&load data

End of week 5 (15 July 2022)

- ☐ Game Design
- ☑ Inital VN for Hunter

- ☑ Inital VN for Hiker
- ☐ Inital VN for Chamois
- ☑ Async for pathfinding
- ☐ Save&Load FogOfWar

Start of week 6 (18 July 2022)

- More save&load data
- Optimisation for mobile performance
- Quests structure and first quest for each character

End of internship (27 July 2022) Optimisations

- ☑ More usage of async
- ☑ DontDestroyOnLoad Map
- ☑ Instance

SaveLoad

- ☑ Save&Load FogOfWar
- ☑ Save&Load First Node for each character
- ☑ Save&Load Envy & Achi
- ☑ Save&Load Quests
- ☑ Restructure and Save&Load DataStorer

Quest

- ☑ Quest & QuestManager class
- ☑ First quest for each character

Finishing touches

- ☑ Game Design
- ☑ UI fixes in general
- ☑ Sorting Layer bug
- ☑ Endgame triggers
- ☐ Better color
- ☑ Exports to Android & Mac