ANKARA UNIVERSITY
ENGINEERING FACULTY
DEPARTMENT OF COMPUTER ENGINEERING

INTERNSHIP REPORT

Game Development using Unity3D & C#

Yeter Tuğba Çetin
16290085

29.07.2019-06.09.2019

ABSTRACT

This report has been prepared for the second of the internships I have worked during the summer term.

During my internship, my role was to develop a mobile-based simulation game using Unity and C#. My goal was to learn the basics of game development on Unity and to move forward on this topic. I also had the opportunity to get information about many other subjects such as modelling, since I was responsible for the project. With the support of my mentor, I managed to complete a great part of the game I worked on in my internship which lasted 26 days.

Ultimately, I became aware of the intricacies and challenges of starting a game project from the rough. I have also gained experience in researching and identifying the most effective way to overcome these challenges.

INSTITUTION INFORMATION

Institution;

Name	: ATOM - Animasyon Teknolojileri ve Oyun Geliştirme Merkezi
Department	: Gyroscoping Games
Address	: ODTÜ TEKNOKENT, Galyum Blok Bodrum Kat, No:15 06800 ODTÜ-Ankara
Telephone	: (+90 312) 987 35 00
E-mail	: atom@odtuteknokent.com.tr
Web Page	: http://www.atom.org.tr

ATOM was established in 2008 as a pre-incubation centre. It provides support to startup teams active in the fields of games and animation, enabling them to take a more active role in the sector.

Pre-incubation centres provide assistance and support services to support potential entrepreneurship and the realization of projects through studies such as technological infrastructure, idea and project support units, branding activities.

Supervisor:

Signature:

TABLE OF CONTENTS

1. INTRODUCTION

I completed my summer internship at ATOM, which is a pre-incubation centre in the fields of animation and game development and located in METU Technopark. My internship started on 29 July 2019 and ended on 6 September 2019 and lasted 26 days. During my internship, I was commissioned to work on a mobile game called Renovate House.

I started my report by giving brief information about the company I work for and by giving a short explanation about their concepts. Then, in the second part of my report, I gave more detailed information about the steps taken according to the project plan. Since the project is planned to be used at a later date, I cannot include all of the source code. However, in-game images and sample code fragments are included.

At the beginning of my internship, there was a problem in the model team of the project we talked about initially. Instead of waiting for the dissolution of the knot in the project, I started this project which has similar features to the mechanics of the other project to spend my short internship period fully productive. The game is based on a player's ability to develop his own home, using the money he earns to move into a home and complete various home repair and decoration tasks to improve it. In the later stages of the project, these houses would be sold and evaluated as multiplayer. However, due to the short period of my internship, this stage could not be started. Since the most important factor in the development of the game is that it is well optimized, I continued to develop the project by updating the sections I developed in line with the new methods I discovered and with the guidance of my mentor.

Following that, I made a sum in the “Conclusion” part of my work and what I have learnt during the internship.

I am very pleased to be included in the atmosphere in this centre. My internship here showed me a lot about entrepreneurship and the challenges of starting something out of nothing.

2. DEVELOPMENT PROCESS

2.1. Creating Database with JSON

A database was needed to buy and sell objects that could be used in the game. That's why files created using JSON hold certain properties of objects. They kept a variety of features, such as specific id numbers, names, categories, 2D images of objects, and file addresses for their prefabs and prices of each object. These values differed according to the use of objects in the game.

The biggest problem with JSON is that it works with arrays, and since the arrays cannot use complex data types while serializing, I have solved this problem by creating an extra class called JsonHelper.

The corresponding prefab of the item selected from the purchase menu was then cloned. The necessary variables were transferred to the class dealing with manipulation for transport and placement functions.

```
[Serializable]
public class Furnitures
{
    public int FurnitureId;
    public string Title;
    public string Description;
    public string IconPath;
    public string PrefabPath;
    public string Category;
    public int Price;
    //x = Resources.Load<Sprite>("Sprites/Furnitures/" + furniture.Iconpath); is used for accessing sprite icons
```

Figure 2.1.1. Furnitures Class for Serialization

```
    void LoadDatabase()
    {
        TextAsset file = Resources.Load<TextAsset>("Databases/FurnitureDatabaseJSON");
        string jsonString = file.text;

        if (jsonString != null)
        {
            Furnitures[] fur = JsonHelperFurniture.FromJson<Furnitures>(jsonString);
            furnitures = new List<Furnitures>((Furnitures[])fur);

            availableFurnitureNumber = furnitures.Count;
        }
    }
```

Figure 2.1.2. Loading Database from Corresponding JSON File

2.2. Creating Player and Mobile Optimization

The player was decided to be the first-person shooter, using a ready-made asset available in the Unity Asset Store. However, this asset did not have mobile optimization. Therefore joysticks are added to the interface for camera movements, jumping and player movements and the functions available in the asset are optimized with an enum for mobile and computer use.

Figure 2.2.1. PC Optimization

Figure 2.2.2. Mobile Optimization

2.3. Object Manipulation

While interacting with the object, various buttons appear on the drop-down panel to sell, fine-tune and position the object. This allows the user to customize the items in the scene as he/she wishes. Object manipulation is achieved by the rays sent by the player around the camera. A ray is an infinite line starting at the origin and going in some direction. If the ray hits an object, the new position of the object updates itself according to these values.

```
///<summary> Manipulate object
///</summary>
private void ManipulateObject()
{
    //objectToManipulate.Layer = 8;          //Layer to manipulationFurn layer
    manipulationFurn.layer = PlayerFurnitureManipulationLayer;          //Layer to manipulationFurn layer
    gameObject.GetComponent<FirstPersonController>().enabled = true;
    var position = manipulationFurn.transform.position;

    if (isMoveFineTune) //moves object for moveunits
        {
            if (isMoveForwardClicked)
            {
                manipulationFurn.transform.position += Vector3.forward * moveUnit;
            }
            if (isMoveBackwardClicked)
            {
                manipulationFurn.transform.position += Vector3.back * moveUnit;
            }
            if (isMoveLeftClicked)
            {
                manipulationFurn.transform.position += Vector3.left * moveUnit;
            }
            if (isMoveRightClicked)
            {
                manipulationFurn.transform.position += Vector3.right * moveUnit;
            }
        }

        if (GetComponent<PlayerInteraction>().RaycastHitOrNot())
        {
            position.x = GetComponent<PlayerInteraction>().RaycastHitPoint().x;
            position.y = GetComponent<PlayerInteraction>().RaycastHitPoint().y;
            position.z = GetComponent<PlayerInteraction>().RaycastHitPoint().z;
            manipulationFurn.transform.position = position;
        }
```

Figure 2.3.1. Raycast and Fine-Tuning Control for Manipulation

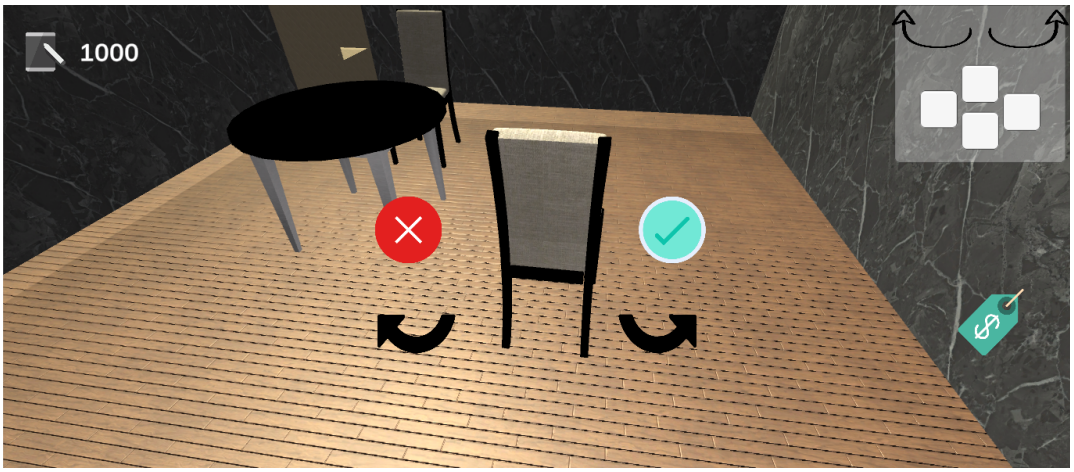

Figure 2.3.2. Manipulation Panel

2.4. Object Interaction

There are various ways of interacting with objects around the player. They are divided into five categories. The dropdown button at the bottom right of the screen provides the switch between these categories.

- **Hand:** In this selection, the player can move the items around by activating the manipulation panel and collect the trash on the floor and empty it into the trash bin. While collecting garbage, the collection period differs depending on the weight of the garbage.

Figure 2.4.1. Collecting Garbage

- **Crack Remover:** One of the deformation types in the house is the various cracks on the walls during the missions or when the player is the host for the first time. To get rid of these cracks, the user must first buy a plaster bucket and then repair the crack. The cracks on which they are repaired are not suitable for painting the wall. Crack remover option should be selected from the dropdown menu. With the help of the resulting trowel, the repair animation starts and the current crack updates the tag as a repaired crack.

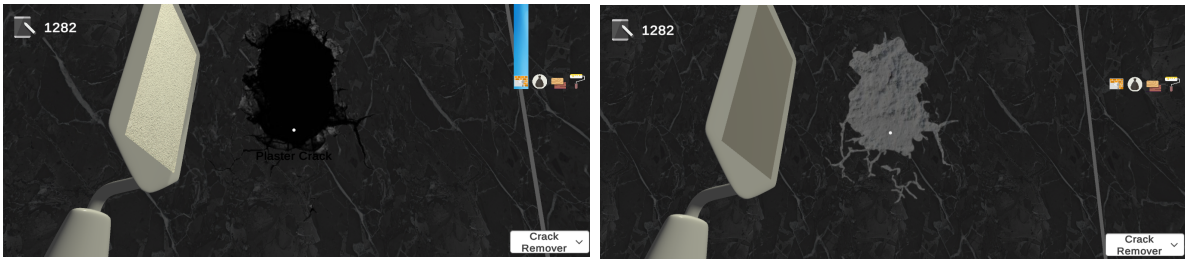

Figure 2.4.2. Before (Left) and After (Right) Plastering Process

- **Painting:** If there are no cracks in the wall to repair, the user selects the colour he likes from the shopping menu and paints the wall with the help of a paint roller. The dyeing process is provided by replacing the material of the surface that is sent to the ray and having the "Wall" tag with the current material in the paint roller. When the painting finishes, the roller returns to its initial colour.

Figure 2.4.3. Before (Above) and After (Below) Painting Process

- **Flooring:** Just like wall painting, floor covering works with the same algorithm, but no tools are used.

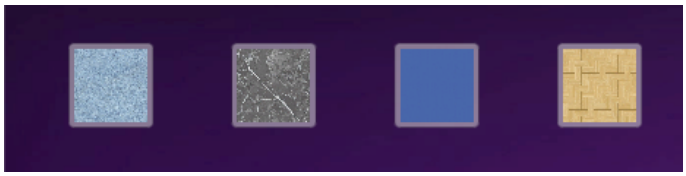

Figure 2.4.4. Flooring Examples from Shopping Menu

- **Cleaning:** Another type of deformation in the game is dust and dirt on the ground. For this, the user selects the cleaning option and completes the cleaning with the broom.

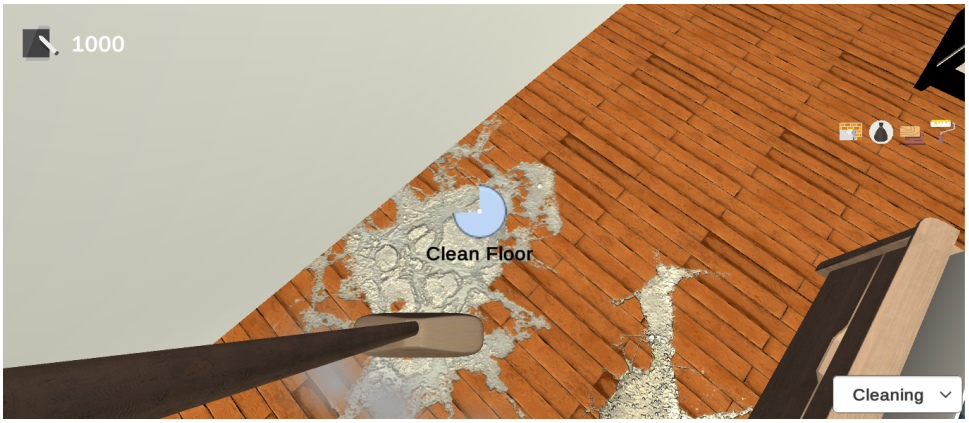

Figure 2.4.5. Cleaning Dirt using Broom with Dust Animation

If all these household goods and other objects purchased are resold, they are sold with an 80 per cent profit if it is a household item, and if it is a used repair item, such as a paint bucket, without any charge.

2.5. Random Object Spawn

There are three types of deformation in the game: dirt, garbage and crack. The positions of these deformations are determined randomly. The number can be adjusted by the developer. In the event of a randomly placed deformation in collision with any object on the scene, it finds and locates the most appropriate random position. Because of both the spawn fields and the variety of deformations, an algorithm with $O(n^2)$ complexity completes this process.

Figure 2.5.1. Garbage Spawn Script on GameManager Object

2.6. Collision Detection

Unity uses colliders to check for collisions between objects. However, as the number of objects increased and I made these ascertainments to examine the overlapping objects on the stage under different conditions and durations with the tags in the scripts I assigned on the objects. The boolean values in these scripts change when the object interacts with the collider of other objects with the correct tags to resolve the collision.

```
public class WallInformation : MonoBehaviour
{
    public bool isPaintable;
    public bool hasPainted;
    void Update()
    {
        foreach(Transform child in gameObject.transform)
        {
            if(child.tag == "Crack")
            {
                isPaintable = false;
                hasPainted = false;
            }
        }
    }
}
```

Figure 2.6.1. Collision Control for Cracks in the Wall

For example, in the code in Figure 2.6.1., if the wall collider is in contact with any crack, the information "Wall cannot be painted." is given to the respective functions.

2.7. Notifications

There are numerous notifications to be given to the user in various conditions in the game. Instead of granting access to the panel and writing the wanted notification as a string text each time, I allowed all declarations to be collected in a single script. In this approach, after activating the panel and assigning the associated text, this script closes the panel within the designated time.

```
private void Update()
{
    if(notificationPanel.activeSelf)
    {
        StartCoroutine(CloseNotification());
    }

    if (notificationTextTablet.gameObject.activeSelf)
    {
        StartCoroutine(CloseNotificationOfTablet());
    }
}
```

```
private IEnumerator CloseNotification()
{
    yield return new WaitForSeconds(2f);
    notificationPanel.SetActive(false);
}

private IEnumerator CloseNotificationOfTablet()
{
    yield return new WaitForSeconds(2f);
    notificationTextTablet.gameObject.SetActive(false);
}
```

Figure 2.7.1. Timer for Notification Panel

2.8. House Value Estimation

The most money-making method in the game at one time is granted by developing and selling the house. Every deformation and old structure inside the house gives a negative influence, while the purchased household goods or renovated parts give a positive influence on the price. Within a specified percentage range of the result, the real estate agent makes an offer to the player. After the sale, the player is directed to the mission screen if he does not have the money to the shopping menu to buy a new home.

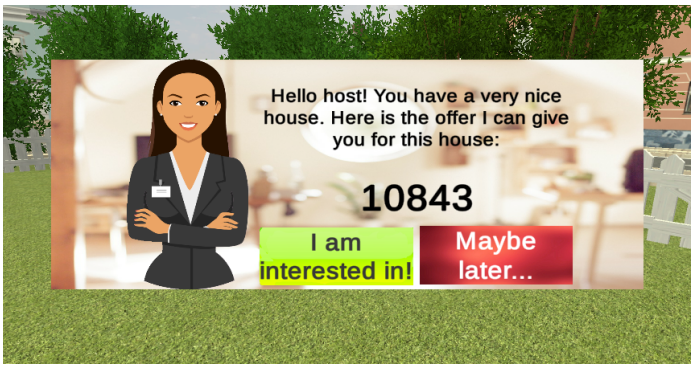

Figure 2.8.1. Conversation with Real Estate Agent

2.9. Mission Generator

The missions in the game are selected from the mission panel on the tablet interface. After each mission has been completed, the player is rewarded according to the difficulty level of the task.

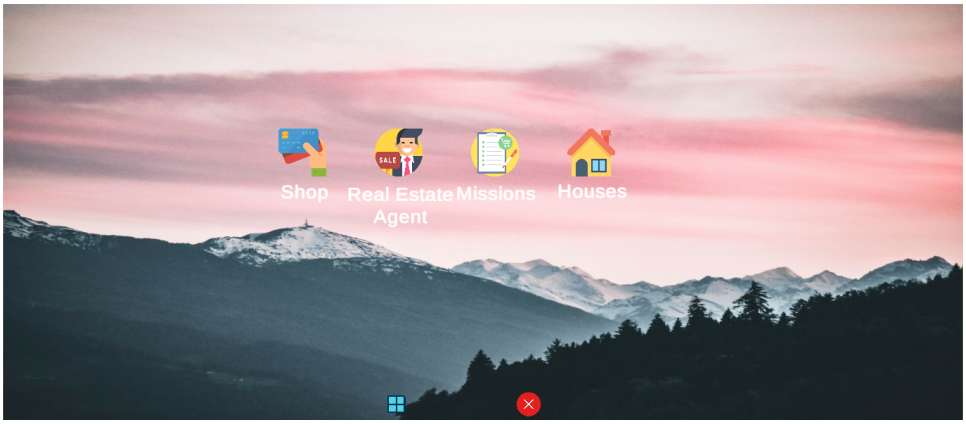

Figure 2.9.1. Player In-Game Tablet Screen

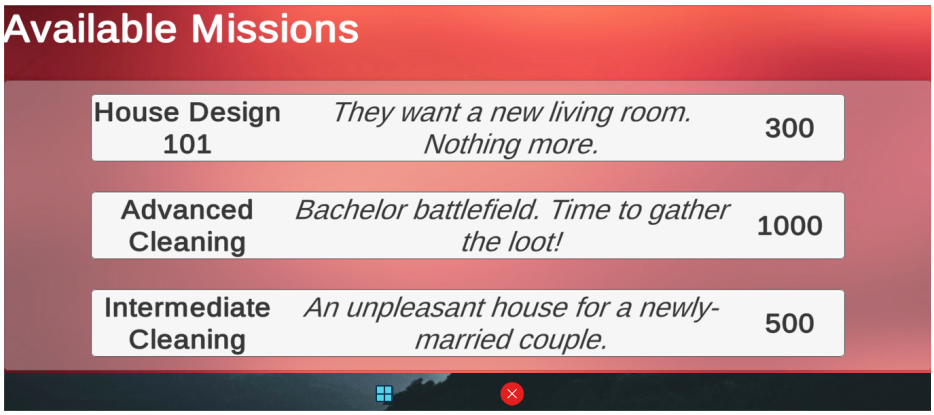

Figure 2.9.2. Mission Screen

Figure 2.9.3. Mission Scene

The tablet screen in Figure 2.9.1. is activated by the icon in the upper left corner of the game screen. The "Missions" button activates the screen in Figure 2.9.2. Hither, three randomly selected missions are displayed from the mission database. The player who chooses one of these missions initiates in the "MissionScene" scene under conditions determined by the difficulty of the mission in a random house prefab. (Figure 2.9.3.) The missions are displayed on the top left corner and when all counters are reset, the mission is completed and the player returns to his/ her house with his award. (Figure 2.9.4.)

Figure 2.9.4. After Mission Completed

2.10. Scenes

During the game development, I created three scenes in the progress I made during the internship period. The first is for the opening screen, the second is for the player's house and the third is for the mission. The start button on the start screen directs the user to the scene of his/ her home.

Figure 2.10.1. Opening Scene

2.11. Modelling

Most of the models in the game I bought from the Unity Asset Store and some web sites. I designed or rearranged some simple materials and models in Blender. I've included these sources in the bibliography.

3. ADDITIONAL PROJECT: ICON STUDIO

The part I had the most difficulty in preparing Renovate House was visual. When creating the purchase menu, I needed 2D images of the prefabs. I was able to edit a small number of prefabs through third-party software. However, this would become very tiring when the database grew and the items needed updating. I developed an asset called Icon Studio to avoid the same problem both in this project and in other projects. Thanks to this asset, it automatically saves prefabs taken from a folder to the specified folders in the desired position, angle and size with the desired background or transparent background with 512x512 pixel size PNG photos.

```
IEnumerator TakeTransparentScreenshot(GameObject sceneObject)
{
    yield return new WaitForEndOfFrame();

    if(objectBackground.GetComponent<Image>().sprite != null)
    {
        Debug.Log("Please pick a solid color.");
        StartCoroutine(ShowMessage("Please pick a solid color."));
    }
    else
    {
        Texture2D texture = ScreenCapture.CaptureScreenshotAsTexture();

        Color[] pixels = texture.GetPixels(startX, startY, width, height);
        Texture2D tex = new Texture2D(width, height);

        for (int i = 0; i < pixels.Length; i++)
        {
            if (pixels[i] == objectBackground.GetComponent<Image>().color)
            {
                pixels[i] = new Color(0,0,0,0);
            }
        }

        tex.SetPixels(pixels);
        tex.Apply();

        byte[] bytes = tex.EncodeToPNG();
        Destroy(tex);

        File.WriteAllBytes(Application.dataPath + "/IconStudio/CreatedTransparentIcons/" + sceneObject.name + ".png", bytes);

        Destroy(objectOnScene);
        prefabIndex++;

        anyObjectOnScene = false;
        isPrefabCalled = true;
    }
}
```

Figure 3.1. Screenshot of the 3D Object with Transparent Background Function

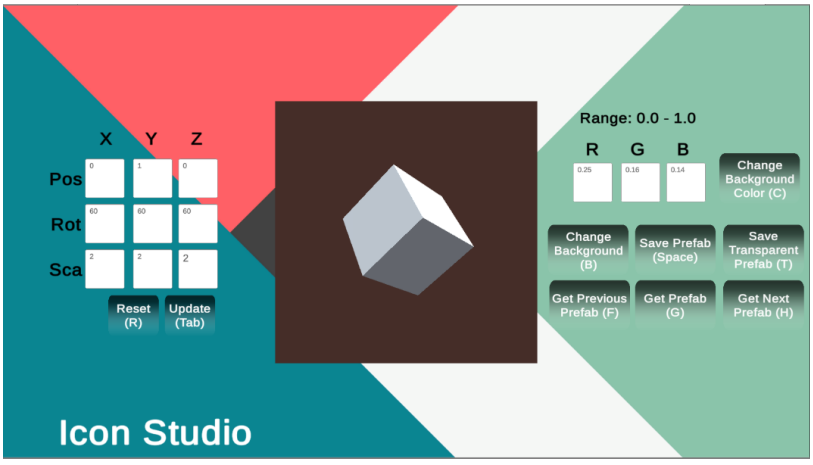

Figure 3.2. Icon Studio Interface

4. USED SOFTWARES

4.1. Unity

Unity is a technology developed by Unity Technologies that is primarily used to develop video games and simulations for computers, consoles and mobile devices. It is a cross-platform game engine that supports 2D and 3D graphics, drag-and-drop functionality, and scripting with C #. It was first announced for MAC OS X at Apple's Worldwide Developers Conference in 2005 and has since been expanded to target 27 platforms.

A convenience Unity provides to game makers is that a game developed with Unity can be compiled on different platforms (PC, Mac, Web, iOS, Android, Windows Phone, Playstation, Xbox, etc.) without any infrastructure changes.

One of the advantages of Unity is that it allows the developer to write program code at the time of game development. Most of the other game engines have separated graphics and code, while Unity and graphics work together. This working philosophy gives the developer flexibility and shortens the development time.

4.2. Blender

Blender is a free 3D modelling and animation software. It is currently financed by donors and sponsors of the Blender Foundation and is developed by two part-time, two full-time working communities at the Blender Institute. It is also one of the well-known and widely used 3D modelling/ design software.

5. CONCLUSION

I have completed my five-week internship at ATOM. The project was a 3D mobile simulation game called Renovate House. The game was built based on a variety of quests, making the player earn money and renovate his home. Most of the models in the game I used free licensed assets and textures on the Internet. I redesigned some incompatible models and created what I couldn't find using the open-source modelling program called Blender. At the end of my internship, I got an idea of the challenges and subtleties of starting game development from the beginning. I've learned more about Unity and C#. I have also discovered new features in Blender, where I have little knowledge.

During my internship period, I received information from my mentor about the plans and methods I should follow that day. At the end of the day, I made a short demonstration of the works I had done. By the time I worked here, the progress made on my personality and work discipline was undoubtedly on my ability to overcome problems on my own. Furthermore, since the project I am working on is a mobile game, I have always paid attention to guarantee that it always performs best in terms of graphics and algorithms I have written. In addition to all this, I have optimized all the assets I use for the game's minimum footprint. To make the project easy to carry, I also followed up unused assets and minimized the size. I have learned how to create a database with JSON that I have never used before and to use it with C# thanks to this project.

Since the thumbnails on the shopping screen in the game will be very difficult to produce and update one by one when the database grows, I have developed an asset called Icon Studio that I can easily get three-dimensional prefabs and PNG previews in the last days of my internship. In this way, the shopping screen has become more sophisticated.

Eventually, I can say that my internship continued under the company's mission. In this pre-incubation centre, I see myself in a status further ahead than before I started my internship. I see that the relationships I have established have improved me socially and professionally. I am very grateful to ATOM for giving me the opportunity to work within them.

6. BIBLIOGRAPHY

https://en.wikipedia.org/wiki/Unity_(game_engine)

https://en.wikipedia.org/wiki/Blender_(software)

https://assetstore.unity.com/packages/essentials/beta-projects/textmesh-pro-84126

https://assetstore.unity.com/packages/3d/environments/nature-starter-kit-2-52977

https://assetstore.unity.com/packages/3d/environments/urban/town-houses-pack-42717

https://assetstore.unity.com/packages/3d/small-town-america-streets-free-59759

https://assetstore.unity.com/packages/3d/environments/urban/simple-corridors-96435

https://assetstore.unity.com/packages/3d/environments/urban/uk-terraced-houses-pack-free-63481

https://assetstore.unity.com/packages/3d/props/exterior/free-stylized-garden-asset-145896

https://assetstore.unity.com/packages/3d/props/furniture/bed-collection-25256

https://assetstore.unity.com/packages/3d/props/furniture/round-carpet-15171

https://assetstore.unity.com/packages/2d/textures-materials/sky/allsky-free-10-sky-skybox-set-146014

https://assetstore.unity.com/packages/3d/props/furniture/realistic-furniture-and-interior-props-pack-120379

https://app.grammarly.com

https://translate.google.com/

https://stackoverflow.com/

https://docs.unity3d.com/Manual/index.html

https://translate.google.com