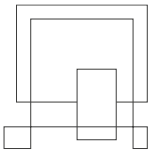


3DVR Challenge

Group FF&

Taha Mohamed Alzein (269055)

Mihai Bogdan Barbus (267082)

Supervisor:

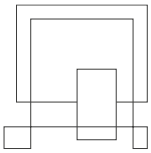
Kasper Holm Bonde Christiansen

Characters 14573

AR/VR Software Engineering, VIA University College, Viborg

4th Semester

10-06-2019



Abstract

The purpose of this semester project is to create and develop one entertaining and challenging system environment experience where players can adapt more easily to the new virtual environment, where players can improve both logical thinking, motion control and increase response speeds.
This VR game experience project was created and implemented with Unity, C# scripts in Visual Studio and use of VR technologies with HTC Vive and Leap Motion.
Related to final product developing team will like to conclude that is an entertaining virtual reality game solution with educational role for children and future

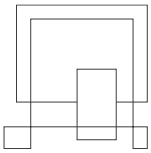
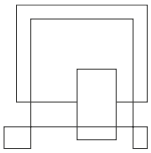


Table of content



1 Introduction

In direct connection with fast evolution of technology in our present times, we can say with no doubts, we already live in the future and in order to keep up with technology, humans should evolve and be better prepared for this digital era. By trying to have an answer to a couple of questions like" How to evolve?" and" How to adjust or prepare humans better for new world?" we discover a challenge in the current times, a new reality, a virtual reality and the main idea is to create a new virtual 3D VR game product with an educational role for adapting all humans in these new technological digital times.

Project is created by a group of two students from Via University College with target group children with ages between 6 to 8 years old.

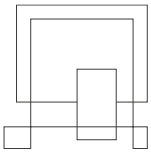
The system is designed with a simple game architecture very similar with MVC design pattern described more in detail at Design chapter, and has one main actor Player, where Player can play a game scene or get result.

By covering all the requirements of this project, our group asses that the project is feasible and brings added quality in the process of e-learning and adapting to new technologies.

2 Idea - creation, elaboration and analysis

Starting from the beginning of the 4th semester our group was informed about the topic for the semester project with a free of choice AR/VR application and we create one plan how to create, elaborate and analyse the idea.

We start with a supervised brainstorming session, where each team member come with three ideas. For each idea we try to make a summary with pros and cons, and we eliminate non-favourable ideas one by one until we reach the most favourable idea for our team.



Chosen group project idea was to create a 3D VR Game to support and encourage the current generation of children, future adults, by familiarizing them with technologies, and developing logical and analytical thinking from small ages.
Considerations on idea project are related to the following themes:

- Educational games

Educational games" are games that are designed to help people to learn about certain subjects, expand concepts, reinforce development, understand an historical event or culture, or assist them in learning a skill as they play. Game types include board, card, and video games"[1]

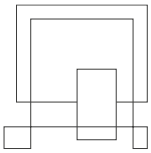
- New technology

New times means new technologies, nowadays we have access to a colossal amount of information with shorter assimilating period and we try to compensate with different methods like" learning by doing".
By creating a virtual game environment, children can experience, upgrading skills and learning by playing in a safe and secure environment as enjoyable as their home

- Puzzles

Puzzles games are regularly used as a tool for development of children's logical and analytical thinking and with the use of new virtual technology we can say its environmentally friendly and according to research "Puzzles for 6-8-year-olds:
At this point, 6 years old, have accrued sufficient skills on jigsaw puzzles. The potential that they have managed to achieve is marked by their ability to define puzzle pieces in relation to its place in the puzzle.
Sorting the puzzle pieces is common in this age group. Each child develops their own unique strategy to problem-solving. They'll do this in accordance to shape, colour, and/or object. Always looking for the next challenge, children in this age group can work with puzzles with up to 260 pieces, depending on their experience, this number can increase up to 500 pieces.
The source of satisfaction lies in choosing the correct puzzle. Bear in mind that the elaborateness of the puzzle is dependent and varies on the individual child's abilities.
Assessing the level of difficulty is important in figuring out the appropriate puzzle game

[1] https://en.wikipedia.org/wiki/Educational_game



for children of all ages. The number of pieces, piece size, the degree of detail of each puzzle, and even the child's experience are all factors that contribute to finding a suitable puzzle to fit the interests of any young child or toddler. Seek information and advice from your pediatrician for any concerns you may have on your child's developmental progress. They understand your child's health and wellness is a priority and are just as happy to help. And as always, don't forget to show tons of support for your child by participating and staying involved as you see fit"[2]

- Living in the future

Indisputable definition like present and future from the technological point of view are obsolete as a result of exponential growth or evolution of technology. We have multiple evidence of this theory and we can add some examples like: Telegraph Video series[3] , Quora topics[4] and even Japanese "future technologies"[5]

Once team reach the decision and one idea was finally chosen, developer team start to elaborate, define and discuss in which way we can develop and add extra value to idea. At this step we try to answer to all possible question like requirements, mechanics and we make a F.A.C.T.O.R analysis shortly presented below.

Functionality-Player assembly a 3D Model for each scene using panels very similar to Lego bricks construction

Application Domain-Entertainment and eLearning for and with Virtual reality experience

Conditions-Game created for 6DoF[6] VR headsets like HTC Vive or Oculus Rift

Technology-Developed with Visual Studio, Unity using Leap Motion hand detect support

Objects-Three different stages with 3D Models

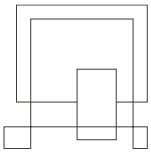
[2] https://en.wikipedia.org/wiki/Educational_game

[3] https://www.telegraph.co.uk/technology/2016/04/25/living-in-the-future-the-livescribe-3-smartpen/?playlist=series:living-in-the-future

[4] https://www.quora.com/Are-we-already-living-in-the-future

[5] https://interestingengineering.com/23-reasons-why-japan-is-already-living-in-the-future

[6] Degree of freedom



Responsibility- Create one really easy user experience, in an appealing way for children

By using for 6DoF HTC Vive player can use the virtual space to move around 3D model he need to assembly and have a 360-degree view of staged 3D Model

For being more accessible to children developers decide to use hands controls in game, and for changing game scenes will be created virtual buttons

3 Requirement & Analysis

3.1 User Stories

1. As a player, I want to be able to move in the scene so that I can change my location.
2. As a player, I want to be able to use my hands so that I can hold bricks using them.
3. As a player, I want to be able to put bricks together so I can assemble a fragmented figure.
4. As a player, I want to be able to detach bricks from each other, so that I can attach them again.

3.2 Functional Requirements

1. The system must allow player to move in the scene.

2. The system must allow the player to use his/her hands in the scene to hold bricks.

3. The system must allow the player to assemble bricks together.

4. The system must allow the player to detach bricks from each other.

6. The system must track time elapsed for each game scene.

7. The system must display on the last scene the time elapsed for each scene and total elapsed time for game.

3.3 Non-functional Requirements

1. HTC-Vive must be used in the system.
2. Leap motion must be used in the system.
3. Unity will be used as a game engine platform.
4. C# will be used as a programming language.
5. The system must be interesting for children between 6-8 years old.
6. The system must have an educational value for children between 6-8 years old.

3.4 Main game scenario

- Play scenario:

Player open game app -> System display Main menu scene -> Player choose to play game -> System display 1st Game Scene and start time tracking-> Player complete 1st scene -> System verify if scene is completed and if true stop timer and display next scene button -> Player choose to play 2nd scene by pressing Next -> System display 2nd Game Scene and start time tracking -> Player complete 2nd scene -> System verify if scene is completed and if true stop timer and display next scene button -> Player choose to play 3rd scene by pressing Next - > System display 3rd Game Scene and start time tracking -> Player complete 3rd scene -> System verify if scene is completed and if true stop timer and display next scene button -> Player choose to go to last scene by pressing Next -> System display Last menu scene with all elapsed time for each scene and total elapsed time for game and display Return button for player to reach the Main menu scene

- Exit scenario:

Player open game app -> System display Main menu scene -> Player choose to quit game by pressing Exit button

3.5 Use case diagram

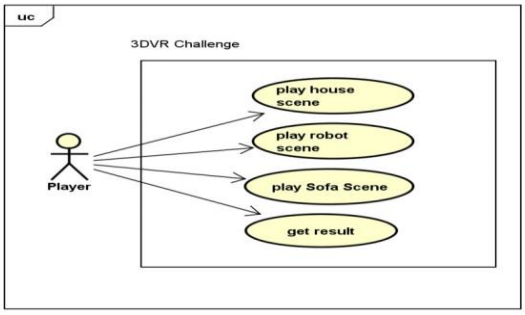


Figure 1 Use case

Use case description

1. Play house scene:
- The Player need to assembly 3D House Models for this scene using hand with polygons anchorable object
2. Play robot scene:
- The Player need to assembly 3D Robot Models for this scene using hand with polygons anchorable object
3. Play sofa scene:
- Player need to assembly 3D Robot Models for this scene using hand with polygons anchorable object
4. Get result:
- The Player press virtual Next button using hands for getting results from last scene

3.6 Activity diagram

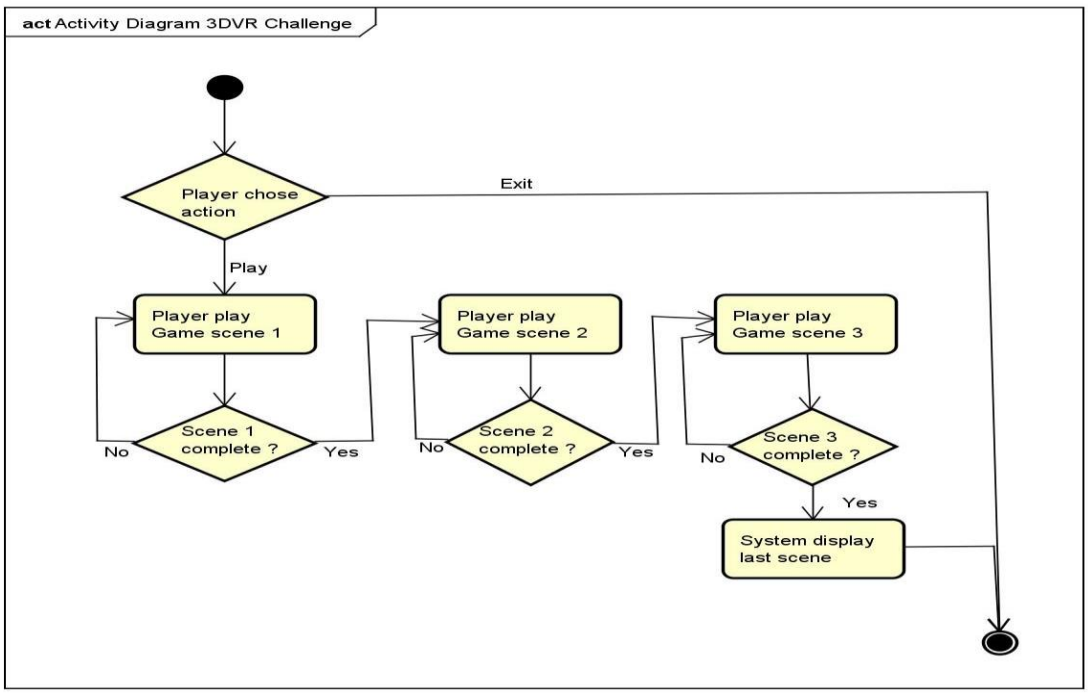


Figure 2 Activity diagram

Activity diagram from above display the flow for the action Play game and show all steps needed for player to play and complete game

4 Design

4.1 Design pattern

Regarding design pattern team fallow a game design pattern like in figure attached below, which is very similar to MVC design pattern with Model, View and Controller, a modular pattern very easy to maintain and add extra features for system

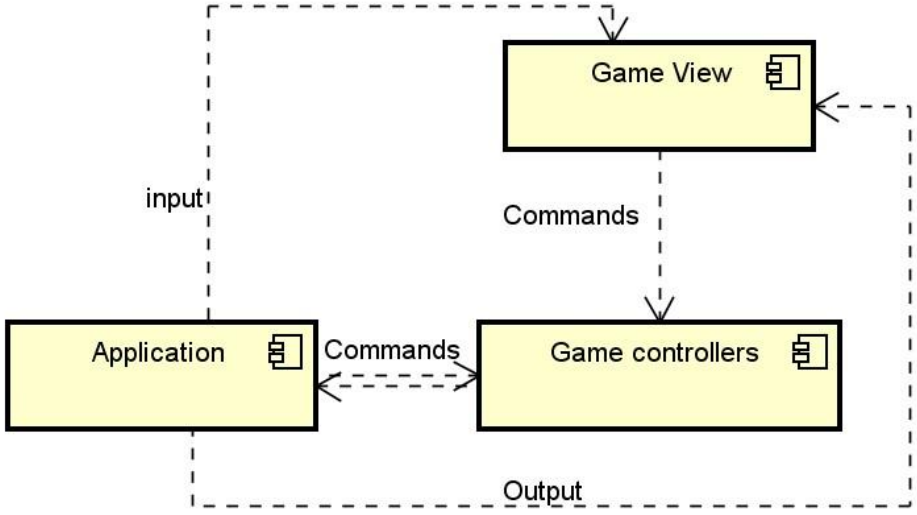


Figure 3 Game design pattern

4.2 Sequence diagram

Sequence Diagrams “are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when”[7].

The sequence diagram below displays the play function, starting with activation in fallowing message sequence input => play => load game scene => game user information.

Sequence fallow all three scene and stop in last scene where just get results like game user information.

[7] https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/

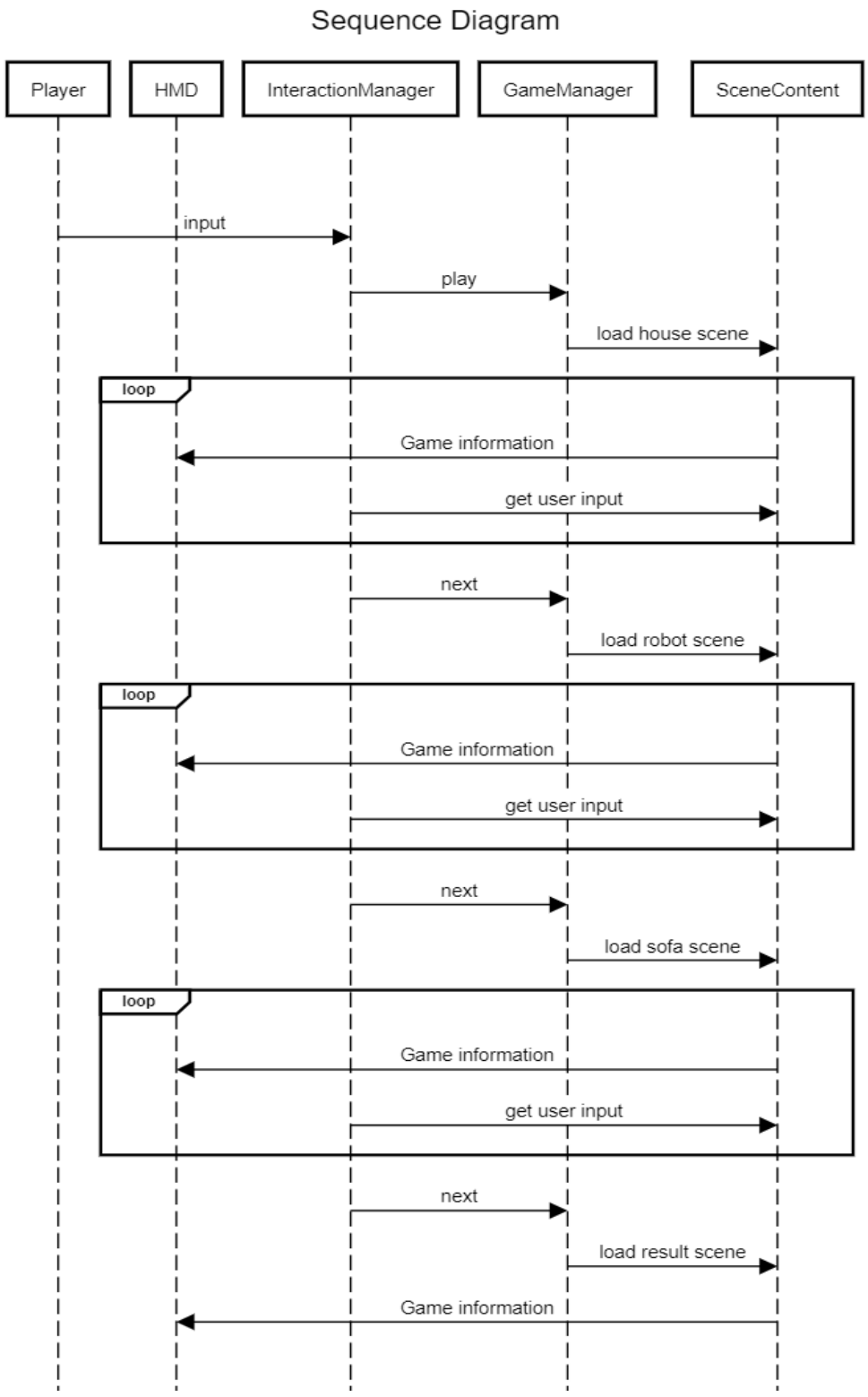


Figure 4 System sequence diagram

5 Implementation

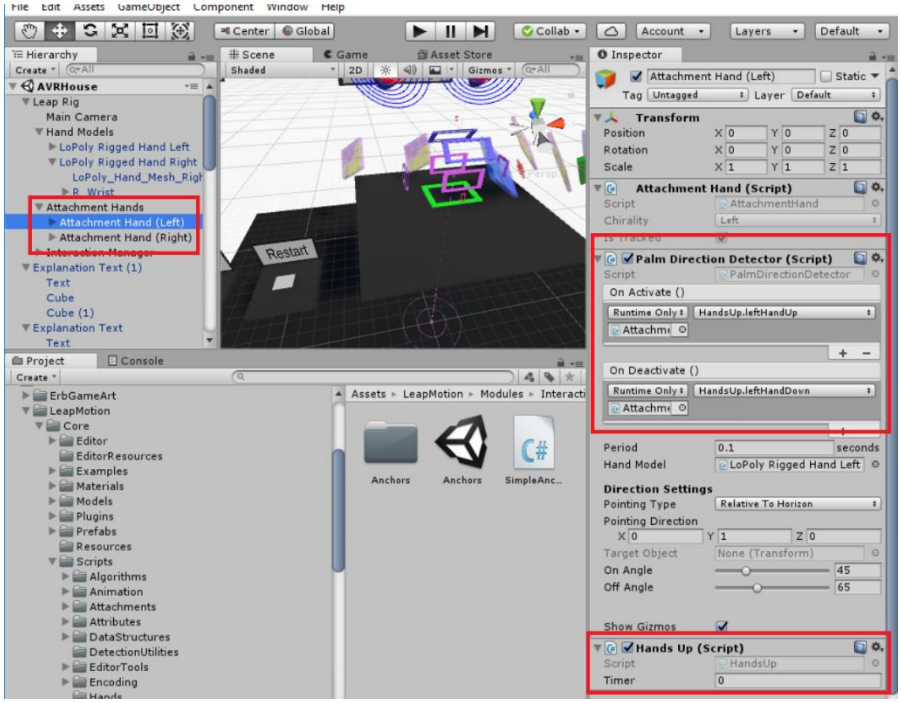
For implementation team use a new system for attaching game objects in order to complete the 3DModels for each game scene, using anchor and anchorable objects.

```
public void allAttached()
{
    Scene scene = SceneManager.GetActiveScene();

    if (scene.name == "AVRHouse")
    {
        if (Magenta1 && Magenta2 && Magenta3 && Magenta4 && Red1 && Red2 && Blue1 && Blue2 && Green)
        {
            AllAttachedBool = true;
            goNextHouse.SetActive(true);

        }
        else
        {
            AllAttachedBool = false;
            timespanHouse = TimeSpan.FromSeconds(Math.Round(Time.time - timer));
            strHouse = "elapsed time" + System.Environment.NewLine + "    " + timespanHouse.ToString();
            goTimerHouse.GetComponent<TextMesh>().text = strHouse;
            goNextHouse.SetActive(false);
        }
    }
    else if (scene.name == "AVRRobot")
    {
        if (Magenta1 && Magenta2 && Magenta3 && Red1 && Red2 && Blue1 && Blue2 && Green)
        {
            AllAttachedBool = true;
            goNextRobot.SetActive(true);
        }
        else
        {
            AllAttachedBool = false;
            timespanRobot = TimeSpan.FromSeconds(Math.Round(Time.time - timer));
            strRobot = "elapsed time" + System.Environment.NewLine + "    " + timespanRobot.ToString();
            goTimerRobot.GetComponent<TextMesh>().text = strRobot;
            goNextRobot.SetActive(false);
        }
    }
    else if (scene.name == "AVRFurniture")
    {
        if (Magenta1 && Magenta2 && Magenta3 && Magenta4 && Red1 && Red2 && Red3 && Red4 && Blue1 && Blue2 && Blue3 && Blue4)
        {
            AllAttachedBool = true;
            goNextSofa.SetActive(true);
        }
        else
        {
            AllAttachedBool = false;
            timespanSofa = TimeSpan.FromSeconds(Math.Round(Time.time - timer));
            strSofa = "elapsed time" + System.Environment.NewLine + "    " + timespanSofa.ToString();
            goTimerSofa.GetComponent<TextMesh>().text = strSofa;
            goNextSofa.SetActive(false);
        }
    }
    else if (scene.name == "AVRLast")
    {

        timespanTotal = timespanHouse + timespanRobot + timespanSofa;
        strTotal = timespanTotal.ToString();
        goTimerTotal.GetComponent<TextMesh>().text = "House " + strHouse + System.Environment.NewLine+ "Robot " + strRobot + System.Environment.

    }
```

```
void Update () {

}
public void resetButton()
{
    Scene scene = SceneManager.GetActiveScene(); SceneManager.LoadScene(scene.name);
}
public void playButton()
{
    SceneManager.LoadScene("AVRHouse");
    Debug.Log("play button is pressed");
}

public void nextButton1()
{

   SceneManager.LoadScene("AVRRobot");


}
public void nextButton2()
{
    SceneManager.LoadScene("AVRFurniture");
}

public void nextButton3()
{
    SceneManager.LoadScene("AVRLast");
}
public void returnButton()
{
    SceneManager.LoadScene("AVRMain");
}
public void exitButton()
{
    Application.Quit();
    Debug.Log("exit touched");
}
```

Above in first script is presented scene complete verification and tracking of time for each scene and total time for all three scenes, and virtual button play in second script.

And here magic happened, in the script below system verify hand position, if both hands are pointing up and whole model is assembled the model start rotation




```
public void leftHandUp()
{

    left = true;
   // Debug.Log("leftHand var is up " + left.ToString());
}
public void rightHandUp()
{

    right = true;
   // Debug.Log("rightHand var is up " + right.ToString());
}

public void leftHandDown()
{

    left = false;
   // Debug.Log("leftHand var is down " + left.ToString());
}
public void rightHandDown()
{

    right = false;
   // Debug.Log("rightHand var is down " + right.ToString());
}

public void NoGravity()
{
    if (left == true && right == true)
    {
        if (AllAttachedBool)
        {

        go.transform.rotation = Quaternion.Euler(90f, 90 * Mathf.Sin(Time.time * 2),  0f);
        //go.transform.position = new Vector3(transform.position.x, 2, transform.position.z);
            }

    }
}
```

6 Test

The focus of our product testing it was on the Acceptance Testing, figure 1, which "is defined as a type of testing performed by the Client to certify the system with respect to the requirements that was agreed upon. This testing happens in the final phase of testing before moving the software application to the Market or Production environment."[8]

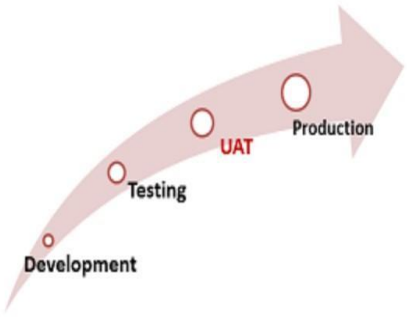


Figure 3 User acceptance test

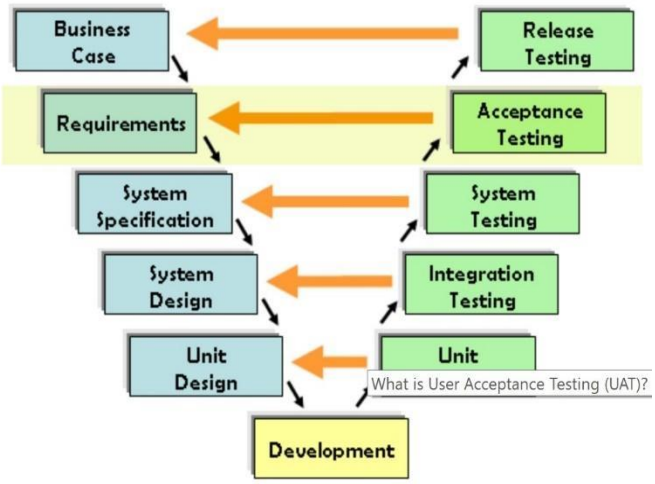


Figure 4 VModel

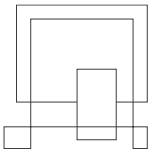
One acceptance test was performed by one girl who is six years old age, that correspond with the targeted group for this educational game targeting children between 6-8 years old persons. During the test was checked all the requirements as in VModel, figure 2, and the features or functionalities and everything work as expected. Based on children feedback we will have a chance to enhance the game in the future with new features like:

- A new slider on side in order to adjust high of game view
- Increase number of 3D Models scene and difficulty of the scenes
- Add extra effect like fireworks and vocal sound guidance

7 Results and Discussion

The outcome of this project is a fully functional 3D virtual reality game which can be deployed on Steam VR. Project can be adapted to different headset technologies and can become a successful commercial version in the close future starting from physical releasing of wireless 3d headsets with Leap motion integrated.

[8] https://www.guru99.com/user-acceptance-testing.html



8 Conclusions & Project future

Developing team agreed that the project fulfill successfully the requirements, and product is complete.

As a future improvement for product we already mention three new features in the test section. From scalability point of view, we can and more game scenes, but we need to take in consideration balance between fun and boring.

From technical point of view we can adjust in the future the product for different emerging technologies like wireless 3D Headsets platforms.

9 Sources of information

- https://skarredghost.com/2017/06/07/need-know-steamvr-tracking-2-0-will-foundation-vive-2/
- https://valvesoftware.github.io/steamvr_unity_plugin/api/index.html
- https://leapmotion.github.io/UnityModules/
- http://blog.leapmotion.com/building-blocks-deep-dive-leap-motion-interactive-design/
- http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/
- https://www.raywenderlich.com/9189-htc-vive-tutorial-for-unity

10 Appendices

- Source code:
- Game manager C# scripts
- HandsUp C# scripts

Appendix A Game manager

```
using UnityEngine.SceneManagement;

0 references
public class gameManager : MonoBehaviour {
    0 references
    public bool AllAttachedBool { get; private set; }

    0 references
    void Start () {
    }

    0 references
    void Update () {
    }
    0 references
    public void resetButton()
    {
        Scene scene = SceneManager.GetActiveScene(); SceneManager.LoadScene(scene.nam
    }
    0 references
    public void playButton()
    {
        SceneManager.LoadScene(sceneName:"AVRHouse");
        Debug.Log(message:"play button is pressed");
    }

    0 references
    public void nextButton1()
    {

       SceneManager.LoadScene(sceneName:"AVRRobot");


    }
    0 references
    public void nextButton2()
    {
        SceneManager.LoadScene(sceneName:"AVRFurniture");
    }

    0 references
    public void nextButton3()
    {
        SceneManager.LoadScene(sceneName:"AVRLast");
    }
    0 references
    public void returnButton()
    {
        SceneManager.LoadScene(sceneName:"AVRMain");
    }
    0 references
    public void exitButton()
    {
        Application.Quit();
        Debug.Log(message:"exit touched");
    }
```

Appendix B HandsUp

```
public class HandsUp : MonoBehaviour
{
    public static bool AllAttachedBool = false;
    public static bool Magenta1 = false;
    public static bool Magenta2 = false;
    public static bool Magenta3 = false;
    public static bool Magenta4 = false;
    public static bool Red1 = false;
    public static bool Red2 = false;
    public static bool Red3 = false;
    public static bool Red4 = false;
    public static bool Blue1 = false;
    public static bool Blue2 = false;
    public static bool Blue3 = false;
    public static bool Blue4 = false;
    public static bool Green = false;


    private GameObject goNextHouse;
    private GameObject goNextRobot;
    private GameObject goNextSofa;



    public static bool left = false;
    public static bool right = false;

    private GameObject go;
    private GameObject goTimerHouse;
    private GameObject goTimerRobot;
    private GameObject goTimerSofa;
    private GameObject goTimerTotal;

    public float timer;
    public static string strHouse;
    public static string strRobot;
    public static string strSofa;
    public static string strTotal;
    public static TimeSpan timespanHouse;
    public static TimeSpan timespanRobot;
    public static TimeSpan timespanSofa;
    public static TimeSpan timespanTotal;

    // Use this for initialization
    0 references
    void Start () {
        go = GameObject.FindWithTag("panel");

        goNextHouse = GameObject.FindWithTag("houseNext");
        goNextRobot = GameObject.FindWithTag("robotNext");
        goNextSofa = GameObject.FindWithTag("sofaNext");

        goTimerHouse = GameObject.FindWithTag("houseTimer");
        goTimerRobot = GameObject.FindWithTag("robotTimer");
        goTimerSofa = GameObject.FindWithTag("sofaTimer");
        goTimerTotal = GameObject.FindWithTag("totalTime");

        timer = Time.time;
    }

    // Update is called once per frame
    0 references
    void Update ()
    {
        NoGravity();
        allAttached();
    }
```

```
0 references
public void leftHandUp()
{
    left = true;
   // Debug.Log("leftHand var is up " + left.ToString());
}
0 references
public void rightHandUp()
{
   right = true;
   // Debug.Log("rightHand var is up " + right.ToString());
}

0 references
public void leftHandDown()
{
    left = false;
   // Debug.Log("leftHand var is down " + left.ToString());
}
0 references
public void rightHandDown()
{
    right = false;
   // Debug.Log("rightHand var is down " + right.ToString());
}

1 reference
public void NoGravity()
{
    if (left == true && right == true)
    {
        if (AllAttachedBool)
       {
           go.transform.rotation = Quaternion.Euler(x:90f, y:90 * Mathf.Sin(f:Time.time * 2),  z:0f);
       //go.transform.position = new Vector3(transform.position.x, 2, transform.position.z);
         }

    }
}

1 reference
public void allAttached()
{
    Scene scene = SceneManager.GetActiveScene();

    if (scene.name == "AVRHouse")
    {
        if (Magenta1 && Magenta2 && Magenta3 && Magenta4 && Red1 && Red2 && Blue1 && Blue2 && Green)
        {
            AllAttachedBool = true;
            goNextHouse.SetActive(value:true);

        }
        else
        {
            AllAttachedBool = false;
            timespanHouse = TimeSpan.FromSeconds(Math.Round(Time.time - timer));
            strHouse = "elapsed time" + System.Environment.NewLine + "    " + timespanHouse.ToString();
            goTimerHouse.GetComponent<TextMesh>().text = strHouse;
            goNextHouse.SetActive(value:false);
        }
```