The Usability of Virtual Reality in On-line Education

Kristína Knapová[1], Gregor Rozinaj[2]
[1] Faculty of Informatics and Information Technologies, Ilkovičova 2, 842 16 Bratislava 4, Slovakia
[2] Faculty of Electrical Engineering and Information Technology, Ilkovičova 3, 812 19 Bratislava, Slovakia
[1] xknapova.stuba.sk, [2] gregor.rozinaj@stuba.sk

Abstract - **The game described in this article was created with the aim to make learning more fun. The game is designed for primary and secondary schools. In general, we can say that technical departments are not attractive to these pupils. That's why we created the game on this topic. We hope that this game will help future college students to decide for the benefit of the technical department. In the game we focused on programming and specifically on the pointers.**

Keywords – Virtual reality; Monster; Level; Pointers; Code

I. Introduction

The programmer's profession may seem tempting to many students. Whether for interesting work or for high valuation. But many will discourage the fact that it seems difficult. Looking into the textbook of programming really discourages many. It does not look as fun as it sounded at first. Our goal is to create a game that will not deter pupils from programming, but their attraction. We wanted to make the game as interesting as possible. That's why we've also used virtual reality. We think it will be interesting for pupils. Our game deals with the theme of the C language. The game translates the pupils with nine topics related to the pointers.

II. Description of the game

The game takes place in tropical rainforest and is divided into nine levels. Each level refers to a part of pointers. In addition, the following scenes are in the game: scene with language, scene with introduction to the game, scene with help, scene with instructions for each level, scene with score gained on current level, scene with labyrinth, and final scene. The game is based on a simple story with a little monster as the main character. The evil inhabitants of the tropical forest abducted monster's two friends. The little monster can do nothing else but go on a journey through the forest and find them. She performs various tasks during her journey. For every accomplished task she gets an instruction about place, where she can find her friends. Tasks, of course, are relate to pointers. The player helps to little monster by accomplishing these tasks. After the last task, she finds herself in the scene where she has to look for kidnapped friends. In this scene she will use the instructions which she received during the journey. Due to this instructions, the little monster can find her kidnapped friends.

III. Description of levels

The theme of the pointers is divided into nine parts in this game. For each part, one level is done.

A. *The program memory*

For anyone who wants to understand the pointer, it's important to understand the program's memory. The first level is dedicated to this topic. To accomplish the task, the player needs to know how the program memory works. Especially, what is saved in which part of memory. At the beginning of the level, the player receives a code sample. He has to remember it so much as possible. He should know, about what the code was. The sample will disappear after a while and the player finds herself on the forest path. It is divided into three parts. Each part symbolizes some piece of the program memory. Its name is written at the top of the scene. There are scattered pieces of code on the path. According to the memory name, the player must specify which pieces of code are stored in it and which are not. Those that are stored there must be taken.

B. *Variable can store address of another variable*

The variable may store the value or the address of another variable. This information is very important for the pointers, because the pointer stores the address of another variable. We want the player to realize that. The second level is therefore dedicated to this topic. At the beginning of the level, the monster finds itself in a small. She may notice that the village is divided into two parts. One part is called the main and the second function. Part names symbolize function names in C language. The function "function" is a function called in the main function of the program. In addition, he will see the river and floating canoe. The canoe will bring the variables to the village. They always fall on the bank of the river. It is the duty of a monster to find the new house for variable. Instructions to find accommodation are written directly on the variable. The monster takes the variable and bring it to its home. There are three options at the house. Whether it wants to accommodate the variable's address, its value or nothing else. This is to symbolize variables, pointers to variables, and the assignment of variables to a function by value or by reference.

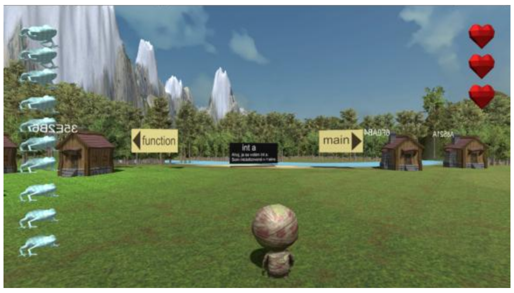


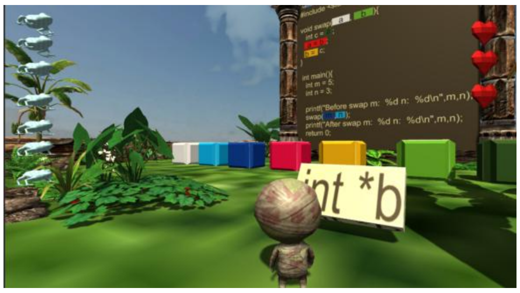
Figure 1.: The memory village

C. *The basic syntax of pointers*

The third level deals with basic syntax of indicators. The level is situated to small meadow. The meadow is full of tables with pieces of code. These tables are answers to questions. These questions are displayed on the big table at the end of meadow. The monster has to find the right answer to question, which she see. When monster hits the right answer, the question is changed to other. Monster has to find the right answer again, till she responds to all questions.

D. *The exchange of variable's values in the function*

In this level, we want to show the usability of pointers. We show it on typical example. In the scene, there is a code, which wants to exchange values of two variables in the function. It is impossible without pointers because the called function makes copies of variable and exchange only them. With the pointers, function can work with these values directly in the memory. The monster finds herself in the labyrinth. The sample code is shown at the end of the labyrinth. Some pieces are incorrect in that code. They has got colorful background. The monster has to find the correct pieces and replace incorrect parts by correct parts.



```
#include [illegible]

void swap( a , b ){
  int c = [illegible];
  [illegible]
  [illegible] c;
}

int main(){
  int m = 5;
  int n = 3;

  printf("Before swap m:  %d n:  %d\n",m,n);
  swap([illegible],[illegible]);
  printf("After swap m:  %d n:  %d\n",m,n);
  return 0;
```

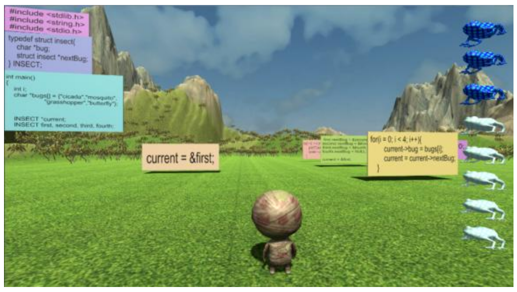
Figure 2.: The monster carries a stone

E. *Pointers and arrays*

Array is a block of memory. The first element of array is also pointer to that block of memory. For this reason, we want to involve arrays among pointers. The monster find herself in an tropical meadow. Animals, which live near the meadow, meet here and prepare some questions for the monster. The questions are about pointers and arrays. The monster has to find correct answer for all of these questions.

F. *Pointers and the structure*

Structures are often used in C programs. They are very powerful with pointers. The many things are implemented with this couple, e.g. linked lists or binary trees. Because of that, we decided to make level dedicated to this theme. The monster finds herself in the meadow again. There is a code, divided to some parts. The goal of code is create structure, create linked list, fill elements of linked list with values and finally print these values. The monster has to collect these parts in the correct order. These parts must be collected in correct order.



```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

typedef struct insect{
    char *bug;
    struct insect *nextBug;
} INSECT;

int main()
{
    int i;
    char *bugs[] = {"cicada","mosquito",
                    "grasshopper","butterfly"};

    INSECT *current;
    INSECT first, second, third, fourth;
```

```
current = &first;
```

```
for(i = 0; i < 4; i++){
    current->bug = bugs[i];
    current = current->nextBug;
}
```

Figure 3.: The monster collect code

G. *Array as argument of function*

An array's address is passed as argument to calling function. This means that the function argument is a pointer to the beginning of the array's block of memory. This enables the called function to change the field values so that they remain altered even after it is terminated. The theme of array as a function argument is introduced in this level. The player is instructed in the level instructions, that the array is created in main function. Then other function is called and an array is passed to it. This function will change the array values. The monster finds itself in front of two rows of stone blocks. One row represents a field in the main function and the other represents a field in the called functions. The duty of the monster is to fill the array in called function with fruit. At the same time, the field in the main function is also filled. After filling monster gets a question as to why the field values changed in the main functions as well.

H. *Dynamically allocated memory*

This part of pointers deal with memory part called heap. This is used, when we need a lot of memory or when we want to have something stored even the function returns. There are several functions for working with heap. In this level we want to introduce them. In this scene is the room with eight pictures with memory state. Samples of function, which work with heap, are scattered around the meadow. The monster has to collect these samples and assign them to memory pictures.

I. *The most common mistakes in pointers*

Pointers is not easy theme for beginning programmer. For this reason we decided to finish this game with common mistakes. This level shows them. The goal of this level is to help start better to beginning programmers. The mistakes are shown in code. The monster has to determine if the code is good or bad. Then she has to hit "good" button or "bad" button. The code is changed after answer.

IV. Description of Scenes

In this game, there are also other scenes, except level scenes. They are described here.

A. The scene with settings

This scene is first and it is very simple. There are just two flags and one button. The player can choose language by clicking to flag. The game is started after click onto flag. The button opens scene with help.

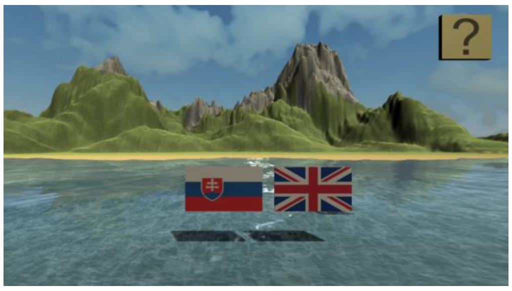


Figure 4.: Flags in scene with setting

B. The scene with help

This scene is very simple. There is just table with help and one button. The player returns to the setting by clinking on it.

C. The introduction scene

The player gets to this scene from setting scene. He can read the story of the game here. Also, he find here first port. He can hit it and he find himself in first level.

D. The scene with instruction to the level

The player can read some instructions about next level here. There is also one button. He can pass to the level by clicking on it. This scene is shown before each level.

E. The scene with results

This scene is opened after each level. The player receive his results here. He also gets an instruction here. This instruction will be useful for him in the final level.

F. The final scene

This is scene, where monster seek her friends. She can help herself with instructions from previous level. She also collect small diamonds. This scene is little labyrinth.

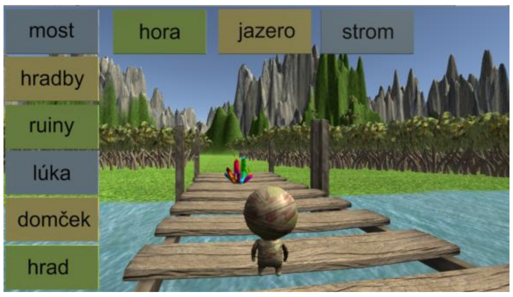


Figure 5.: The monster goes to pick a diamond

G. The winning scene

This scene is ultimate scene. There are happy two saved monsters with their happy savior.

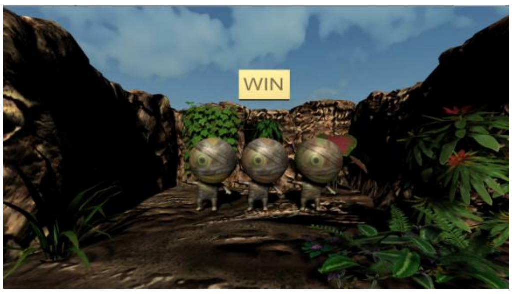


Figure 6.: Happy monster are together

V. IMPLEMENTATION

There are several background functionalities, which we want to mention. Except this, we want to mention used platforms and environment.

A. Background functionalities

This is group of functionalities, due which this game works.

1) The level loading

Each level has its number. In the each level, there is one object called LevelBegin. This object has script SetLevelId and this script has a public variable. The level number is set to this variable. The level number is used when the instruction level or result level is loaded. Due to this number, they know which data they have to load. The script SetLevelId uses singleton construction. Due this, it can provide and change this variable in many scenes and all these scenes can see the changes.

2) The result manager

The result manager count all points which are gained in levels. It stores the total number of coins an manage the rules for levels. Player has three lifes. It provide four changes for him. If he finish level with less than four of them, the script StateFrog activate object LastPort and the player can move to next level. If he make four mistakes, the level restarts. The good and bad answers are counted on the different places in scripts. The player can see the state of his answers on the left. There is "frog bar", which shows him the count of answers. The script called *StateFrog* takes care about all of that.

3) The language setting

The translation to languages starts in scene with settings. There is an object called flags. It is parent object for flag's objects. It has several scripts. All of these scripts do the same. They are made for each level which has textures with some text. They take these textures to the array for several languages. We have implemented it for two languages now. These scripts stores textures and have method which returns the array with textures. It returns them depending on choosed language. The next used scripts are EnFlag and SkFlag. They are children of the object flag. They takes his scripts which stores and return textures. The make objects from them. We can call them t-objects. When one of flags is fired, the script EnFlag or SkFlag starts, depends on choosed flag. T-objects take textures of choosed language and return them. Their return values are set to another method. This method is from Language class. It sets class textures variables to its arguments. So texture variables of class Language are set to textures of choosed language. This class is realized by singleton construction. The communication between language „setters" and language „getters" works by this way. The class Language sets all texts and textures to the desired language and provide them to levels. Each scene which needs translate some texture has a script PaintTexture. This script belongs to the LevelBegin object and has the public array of GameObjects. This array takes all objects which have to be translated. The script aslo has object of Language class. This object provide textures for these GameObjects. The translation is made by following way. At first script gets the level number. Then it determine which objects have to be textured due to the level number. The script takes this objects and asks about textures for the actual level. Then just paint textures to the objects. The text translation is similar. In the Language script, there are texts in two languages for each text output in the game. These languages are extensible. Each level, which needs to translate text, has got its own script to do this. These scripts are similar. They have got public variables for taking text GameObjects. They have got an object of Language class. Methods of this class are used for set text in choosed language. The Language class know the choosed language and sets the text based on this choose.

B. Used platforms

The game has to work in two platforms. It runs as desktop game under Windows and it runs as virtual reality game under Android. It is decided under which platform game will works before game built. Differences between Android and Windows versions are minimal. Google cardboard sdk features are used when it is built for android. There is special player for virtual reality and tiles for moving in virtual reality too. There are special conditions which determine what platform is used. Features for virtual reality or for the desktop are activated depends on this conditions.

C. Enviroment

The game is developed in Unity 3D [1]. The most of game objects comes from Unity Assets Store [2]. Remaining game object are default Unity game objects. Textures was created in the Corel Draw X5. Several textures come from the internet. These textures are pictures of animals or pictures of fruit. The Google cardboard sdk [3] si sused for virtual reality.

Acknowledgment

The research described in the paper was financially supported by the H2020 project NEWTON, No. 688503 and VEGA project INOMET, No. 1/0800/16.

References

[1] Unity. 2017. Unity. https://unity3d.com/

[2] Unity. 2017. Unity. https://assetstore.unity.com/

[3] Anonym. 2018. Anonym. https://developers.google.com/vr/develop/unity/get-started