Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Engineering

Bachelor Thesis

Multimedia application in C#

Petr Juditka

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

BACHELOR THESIS ASSIGNMENT

Petr Juditka

Informatics

Thesis title

Multimedia application development in C#

Objectives of thesis

This thesis is dedicated to development of multimedia applications using C# programming language. The main goal of the thesis is to design and implement an educational game. The subgoals are to describe technologies and environments available and select the most suitable one for such a task.

Methodology

The methodology of the thesis is based on study of technical and scientific sources related to multimedia application design using C# language. Based on the synthesis of the gained knowledge all the available technologies and development environments will be described. Then the most suitable one for completing the practical part will be selected.

The practical part of the thesis will concern analysis, design and implementation of a multimedia application. The application will be an educational game for practising calculations. The application will be deployed, tested and evaluated. Based on the evaluation the possible changes and improvements will be proposed.

The proposed extent of the thesis

35-40 pages

Keywords

C#, MonoGame, XNA, OOP, .NET, MathGame

Recommended information sources

Michael Fleischauer, Cross Platform Game Development with MonoGame, 2015, ISBN 9781310496943
Rob Miles, Introduction to programming through game development using microsoft xna game studio, Microsoft Press 2010, ISBN-13 978-0735627130
Simon Kendal. Object Oriented Programming using C# . : Dokboon, 2011. ISBN 978-87-7681-814-2.
Svetlin Nakov & Co. Fundamentals of computer programming with C#. 2013. ISBN 978-954-400-773-7.

Expected date of thesis defence

2016/17 SS – FEM

The Bachelor Thesis Supervisor

Ing. Jiří Brožek, Ph.D.

Supervising department

Department of Information Engineering

Electronic approval: 1. 11. 2016

Ing. Martin Pelikán, Ph.D.

Head of department

Electronic approval: 1. 11. 2016

Ing. Martin Pelikán, Ph.D.

Dean

Prague on 14. 03. 2017

Declaration

I declare that I have worked on my bachelor thesis titled "Multimedia application in C#" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the bachelor thesis, I declare that the thesis does not break copyrights of any their person.

In Prague on 15. 3. 2017 _____

Acknowledgement

I would like to thank Mr. Brožek and my colleagues, for their advice and great support during my work on this thesis.

Multimediální aplikace v C#

Souhrn

Tato práce se zabývá problematikou návrhu a vývoje multimediální aplikace v rozhraní .Net za použití programovacího jazyka C#. Hlavním cílem bude udělat návrh naučné hry, ve které bude hráč procvičovat výpočet základních matematických operací. Dále bude tato hra implementována, vyzkoušena a na základě výsledku bude navrhnut možný vývoj do budoucna. Sekundární cíl bude zahrnovat analýzu použitých technologií – .Net, rozhraní MonoGame. Práce bude také zahrnovat problematiku základního herního návrhu.

Klíčová slova: .NET C# MonoGame XNA OOP

Multimedia application in C#

Summary

Following work will be dedicated to multimedia application using .Net framework and specifically C# programming language. Main goal will be to design and implement educational game – using MonoGame framework – which could be used for encouraging students of mainly elementary schools into practicing mathematical skills. I will also focus on fundamentals of game development and theory behind. I will do briefly introduction into MonoGame framework and what it provides for programmers – basic namespaces with classes, structures and functions.

Keywords: .NET C# MonoGame XNA OOP

Table of content

List of figures

1 Introduction

Following work will be dedicated to multimedia application using .Net framework and specifically C# programming language. Main goal will be to design and implement educational game – using MonoGame framework – which could be used for encouraging students of mainly elementary schools into practicing mathematical skills. I will also focus on fundamentals of game development and theory behind. I will do briefly introduction into MonoGame framework and what it provides for programmers – basic namespaces with classes, structures and functions.

2 Objectives and Methodology

2.1 Objectives

This thesis is dedicated to development of multimedia applications using C# programming language. The main goal of the thesis is to design and implement an educational game. The sub goals are to describe technologies and environments available and select the most suitable one for such a task.

2.2 Methodology

The methodology of the thesis is based on study of technical and scientific sources related to multimedia application design using C# language. Based on the synthesis of the gained knowledge all the available technologies and development environments will be described. Then the most suitable one for completing the practical part will be selected.

The practical part of the thesis will concern analysis, design and implementation of a multimedia application. The application will be an educational game for practicing calculations. The application will be deployed, tested and evaluated. Based on the evaluation the possible changes and improvements will be proposed.

3 Literature Review

3.1 .NET

It is "a general purpose development platform for any kind of app or workload, providing key capabilities for building high quality apps including automatic memory management and support for modern programming languages"[1]. Basically .Net consists of four parts – programming language, visual studio, virtual machine (CLR) and libraries.

3.1.1 Programming language

In my case programming language will be C#. Microsoft defines C# like an "elegant and type-safe object-oriented language that enables developers to build a variety of secure and robust applications that run on the .NET Framework"[2]. With C# you can create a many various kinds of application from windows client applications, XML web services, to database applications and much more. C# also relieves programmers from some work through taking care of occupied memories which won't be used anymore. Unlike in C/C++ where you need to take care of this yourself, in C# there is feature called Garbage Collector which does this routine for us.

C# is one of languages with so called virtual machine. Code is at first compiled to CIL – Common Intermediate Language. This is basically machine code but with slightly simpler list of instructions and also is directly supporting OOP – Objected Oriented Programming. In next step is this machine code interpreted to processor using interpreter – in .Net it is Common Language Runtime (CLR). This operation is hastened because of our code is already in more simple format. The whole process you can see on the next figure. (Čápka)

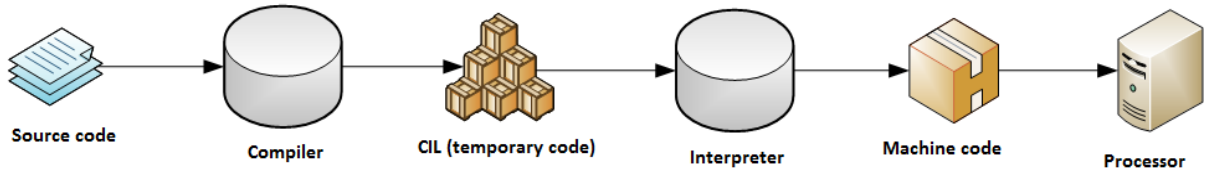

Figure 1 - Compile process of C# (based on source of chapter)

[1] Microsoft's .NET definition - https://msdn.microsoft.com/en-us/vstudio/aa496123

[2] Microsoft's C# definition - https://msdn.microsoft.com/en-us/library/z1zx9t92.aspx

3.1.2 Integrated Development Environment

There are more options when choosing game development environment. There are two basic types of approach. First one is low level when you need to code all things like physics and other stuff by yourself. One of examples of such environment is visual studio. On the second hand is standing high level. Environments with this approach are usually called also engines. In engine you have already prepared entities events and other stuff with some default settings. So you can rather use already done stuff or slightly modify to your preferences.

Figure 2 - Visual studio logo (source: https://www.visualstudio.com/)

Visual Studio fits into the first group of environments I mentioned – low level. Although programmer needs to do a lot of programming, Visual studio is a great tool for programming with many features from Intellisense which really speed up your coding, error list which is great help when dealing with annoying issues, up to many refactoring methods which saves you lot of coding. One more feature that I would like to mention is run-time debugging which comes handy in case you are looking for logical issues hidden within your code.

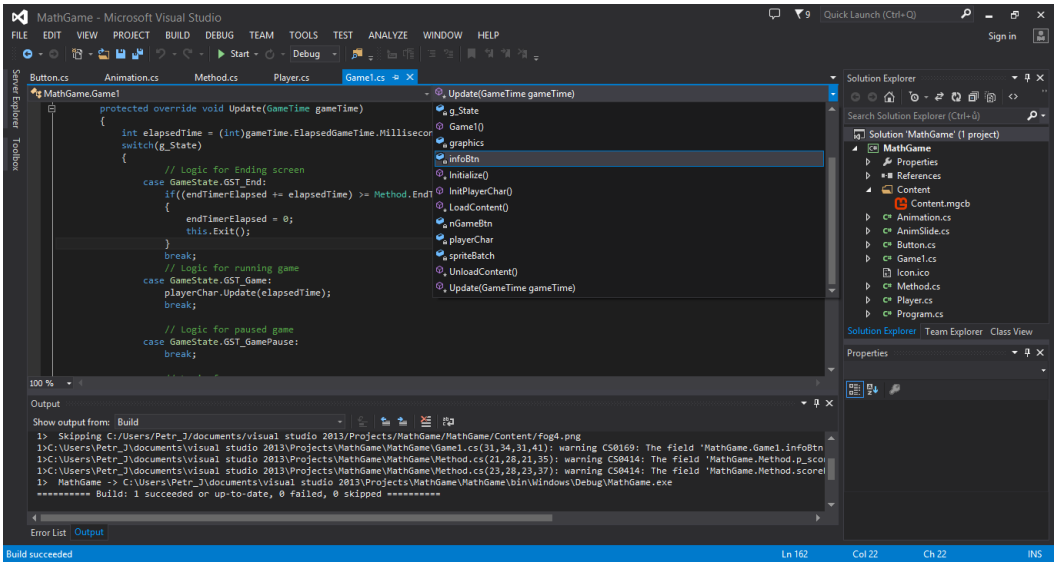

Figure 3 - Visual Studio Interface (self authored)

Figure above is example of Visual Studio interface. As for the layout you can adjust to your preferences. There is lot of windows and views which can be added. On the example you can see solution explorer on the right side which depicts structured solution(s) with their own files. In the lower part is shown console output, in this part IDE is displaying messages from various functions like compiling the solution. You can also send messages from own code in order to analyze its behavior in case of logical errors in your program.

3.1.3 Virtual machine

In .Net case the virtual machine is already mentioned CLR – Common Language Runtime. Which takes precompiled objected oriented machine code – CIL. And interpret it as a machine code for our physical processor.

3.1.4 Libraries

On of biggest advantage of .Net are libraries which support us with predefined set of components and functionality. These libraries contain tools for various tasks from console, database or form applications and lot of more. Because Microsoft is also author of Windows operating systems each component is well designed in order to be in harmonize with the system. In order to application work, on end user's computer there need to be installed same version of .Net like the one it was developed with.

3.2 Game development fundamentals

Since my final solution will be situated in 2D I will be only mentioning basics of 2D games development. But let us start with principle which all games shares in common – basic game loop. First we read input then update state of the game dependent on input. Afterwards are made checks for various conditions – collision, victory, timer etc. And the last step is to refresh game screen. (Howland)

Now let us move over to graphics. 2D coordinate system which is used in games measures whole window in pixels from the top left corner. These coordinate are used for most of the drawing.

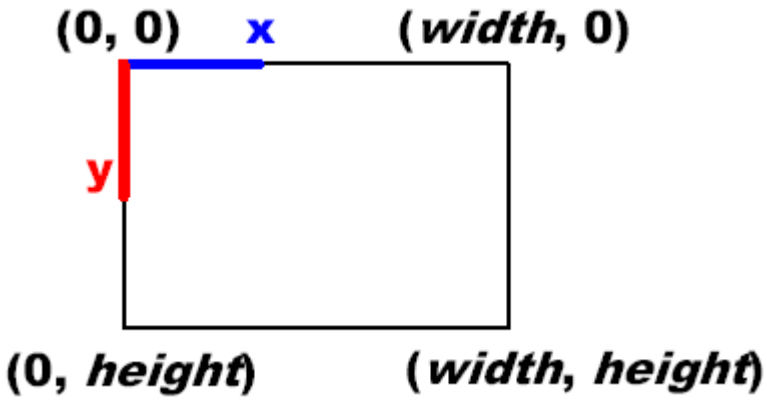

Figure 4 - 2D coordinate system (source: http://rbwhitaker.wikidot.com/monogame-introduction-to-2d-graphics)

As you can see on figure 4 the system's origin is allocated in top left corner from where are starting x-axis and y-axis. While x-axis goes right till the full width of window, y-axis is directed down and ends on window's height.

In computer graphics a sprite is two-dimensional image. Since we are in 2D coordinate system most of our graphics will be about drawing some sprite on given coordinates. (Whitaker)

3.3 MonoGame framework

Figure 5 - MonoGame logo (source: http://www.monogame.net/)

MonoGame is based on the original Microsoft developed XNA framework which was declared as obsolete after some time and its development was stopped. Although the name is different now, the core is the same. After Microsoft announced the retirement of XNA another framework raised open source project XNA Touch which has goal to enable 2D XNA games to be run also on mobile devices. Subsequently the project was published on the Github site and its name was altered to MonoGame. Later on project was enriched by adding support for more platforms which were Android, Mac, Linux and OpenGL on Windows. (Spilman)

MonoGame provides basic functions for updating game logic and drawing components on screen. Also it comes with many classes and structures in order to soften up programmers' job while creating mostly 2D games. Framework itself contains a lot of different namespaces which consists of various classes, enums and structures. (Fleischauer, 2015)

The most fundamental classes are in the "Microsoft.Xna.Framework" namespace. This includes the class for game itself which provides us important functions for initializing game variables, updating logic and drawing method. Besides game class there is a lot of more classes such as GameTime which provides us timer of between each steps either from the real start of application. Another important things are structures – data types – such as Vector2(3) for positioning on the screen which also provides lot of functions for multiple operations with vectors. When position alone is not enough there is also Rectangle structure which upgrades 2D vector higher by adding width and height. Rectangle is also supporting us with various of handy functions, for example checking whether two rectangles collide with each other.

Next important namespace which MonoGame contains is "Microsoft.Xna.Framework.Graphics". And it consists of classes crucial to drawing on the screen. One of these classes is "SpriteBatch" which enables a group of sprites to be drawn using the same settings (Microsoft). SpriteBatch consists of functions for drawing not only sprites but also can draw text. It also provides us a class for sprites – Texture2D(3D) so we have a way for to store and draw them later. And for drawing text there is class „SpriteFont" which can represents font for drawing texts on the screen. It also provides functionality of measuring text in given font to precisly calculate its desired position.

3.4 Expression parsing algorithm

Since player won't be only one to calculate mathematical expressions but the game itself will have to calculate it to in order to check whether chosen answer is correct. We need some way to calculate expression from text form. For this reason we can use algorithm described by (Tan) on dream in code website. This algorithm consists of few steps. At first we transform regular expression into the Reverse Polish Notation also called Postfix notation. As it prompt the RPN in short is inversion of "Polish Notation" which is called Prefix notation. Prefix notation puts all operators before their operands. In case of RPN as it prompt is vice-versa. On the Figure below you can see the example of expression within Prefix and Postfix notations.

```
Regular expression:   1 + 5 * 2

Prefix notation:      + 1 * 5 2
Postfix notation:     1 5 2 * +
```

For converting of regular expression – also called infix notation – we can use Shunting yard algorithm. In order to better understanding at first you can look at the figure on next page which contains first piece of algorithm itself in detail.

```
While there are tokens to be read:
   Read a token.

   If the token is a number, then add it to the output queue.

   If the token is a function token, then push it onto the stack.
```

Basically it takes every token one after another and make use of two types of memory –queue and stack. After reading a token there is check for what type of token it is. Numbers are send to the output queue – memory based on FIFO (first in – first out). On the other hand operators are stored within the stack – memory based on LIFO (last in –first out). If token is either number or function, algorithm continues with checking of other token types.

```
If the token is a function argument separator (e.g., a comma):
     Until the topmost element of the stack is a left
parenthesis, pop the element onto the output queue.
     If no left parentheses are encountered, either the separator
was misplaced or parentheses were mismatched.

If the token is an operator, o1, then:
     while there is an operator, o2, at the top of the stack, and
either
          o1 is associative or left-associative and its
precedence is less than (lower precedence) or equal to that of o2,
or
          o1 is right-associative and its precedence is less than
(lower precedence) that of o2,
          pop o2 off the stack, onto the output queue;
       push o1 onto the operator stack.
```

tokens from stack before left parenthesis. In case left parenthesis is not found within the stack it will finish with error – either the separator was misplaced or parentheses were mismatched. Next it test token whether it contains operator. If so algorithm needs to assure that last added token to the stack is not operator with higher precedence then our current token. In such case the last token will be send to the output queue. Independently on the previous check the current token is added to the stack of operators.

```
If the token is a left parenthesis, then push it onto the stack.

If the token is a right parenthesis:
       Until the token at the top of the stack is a left
parenthesis, pop operators off the stack onto the output queue.
       Pop the left parenthesis from the stack, but not onto the
output queue.
       If the token at the top of the stack is a function token,
pop it and onto the output queue.
       If the stack runs out without finding a left parenthesis,
then there are mismatched parentheses.
```

In case the token is parenthesis algorithm will either put it on the stack if it is left parenthesis. In the other case it will pop off every operator until it finds left parenthesis on top of the stack. In case left parenthesis is not present it will end with error like with the function argument separator.

```
When there are no more tokens to read:
   While there are still operator tokens in the stack:
       If the operator token on the top of the stack is a
parenthesis, then there are mismatched parenthesis.
       Pop the operator onto the output queue.
Exit.
```

This should cover all possibilities which could happen. However there is one final step which still needs to be done. Whether there are some operators left in the stack they should be sent to the output queue. Now the procedure of converting regular expression to postfix notation is complete and the result can be found in the output queue.

This was only first step of the entire algorithm. Shunting yard algorithm gives us expression in postfix notation but it still need to be calculated. For calculation there is RPN evaluation algorithm which basically go through previously filled queue and for each token is testing whether it is value or a function. In first case token is simply pushed to the stack. In case of function algorithm will pop the top n values from the stack if there is enough values. Evaluate given function with values taken from the stack as arguments and the result will be pushed back into stack.

```
While there are input tokens left
   Read the next token from input.

   If the token is a value
         Push it onto the stack.

   Otherwise, the token is a function. (Operators, like +, are
simply functions taking two arguments.)
         It is known that the function takes n arguments.
         So, pop the top n values from the stack.
               If there are fewer than n values on the stack
                     (Error) The user has not input sufficient
values in the expression.
         Evaluate the function, with the values as arguments.
         Push the returned results, if any, back onto the stack.
```

stack. In case there is more than one value algorithm should end with an error – expression contains too many values. In case everything went well there is only one value in the stack – final result. (Tan)

```
   If there is only one value in the stack
         That value is the result of the calculation.
   If there are more values in the stack
         (Error) The user input too many values.
```

4 Practical Part

4.1 Choosing IDE

My final choice is the only lower level from my list, Visual Studio with MonoGame framework. Main reasons were that I am used to Visual studio graphical interface and also that I quite liked the NXA framework which is root for the MonoGame. Some more things which also played a part in my conclusion was that by using lower level environment is good for learning how it works. Also that I am not really used to interfaces and proceeding in other possibilities. Lastly I would like to mention that this is personal choice based on my own preferences. So this doesn't mean Visual studio is necessarily better environment than others.

4.2 Game specification

Result of my bachelor work will be application which could be also called mathematical game. In this game player will control main character and its goal will be to get so far as possible. But without obstacles it would meaningless, so on his way player will meet line of various blocks. Each of these blocks will have its own number, possible result to formula which will be generated and displayed on the screen. One more obstacle for player will be time. For calculating and going over the right block player will be given certain time interval. In case not going over one of blocks during this time player will be caught in deadly element (it will vary dependent on background, for example: fire, hurricane, flood, etc).

There will be possible to choose various themes in settings which will be available in game menu. Dependent on selected themes will be generated mathematical expressions. Also will be possible to select difficulty level which will influence numbers in expressions (higher difficulty - higher numbers, decimal places) and time interval for each question.

In order to players could compete against each other game will consist of High Score system. Players will receive points for each correct answer he chooses. Points given for single question will differ dependent on how much time remains and also on selected difficulty level. Players with best results will be saved in High Scores.

4.3 Interface design

In this part I will describe user interface concept of my application. Application will consist of menu, settings, high scores and the most important the game screen. Let us start with screen user will be shown the first – menu.

4.3.1 Menu screen

For the whole application menu is kind of crossroad which will navigate user through. Screen will consist of four buttons – new game, settings, high scores and exit. These buttons will be centered to screen as they are only objects in the menu screen. On this screen whole functionality will handle mouse. On figure below you can check how the screen will be designed.

Figure 6 - Menu screen design (self authored)

4.3.2 Settings

This screen will contain text boxes with settings which will influence the game itself. Parameters which user will be able to change will be difficulty level and used operators for mathematical expressions. In order to change some parameter user will have to click on given text box to change its value. This screen will be mainly controlled by mouse although there is also a keyboard function – return to the menu by pressing escape. Layout of settings screen you can see on the figure below.

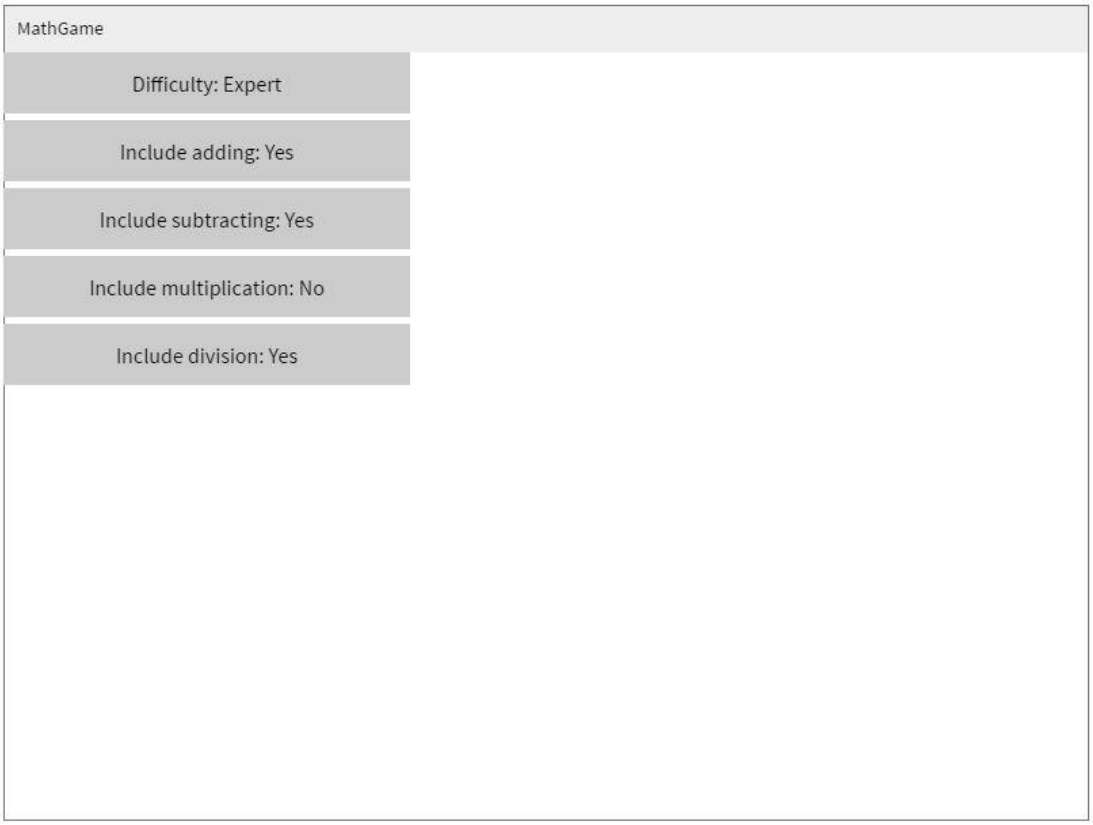

Figure 7 - settings screen design (self authored)

4.3.3 High scores

This screen will be dedicated to recording the best ten scores players achieved. Main and only layout will consist of ten text boxes which will represent the very best scores. As it was at the menu also here the text boxes will be centered to the screen. There is any use of mouse however there is same keyboard function like in the settings – return to the menu by pressing escape. On the figure below you can go over the layout design for high scores screen.

Figure 8 - high scores screen design (self authored)

4.3.4 Game screen

The final screen and also the bed-rock of the application will consist of various elements – player character, blocks with possible answers and also few text boxes displaying valuable information such as mathematical expression to be solved or score player achieved so far. During game player won't need to use a mouse unlike on the other screens but there will be whole control provided by keyboard. Before I will start describing graphical layout of the game screen there is another figure below which shows the layout design.

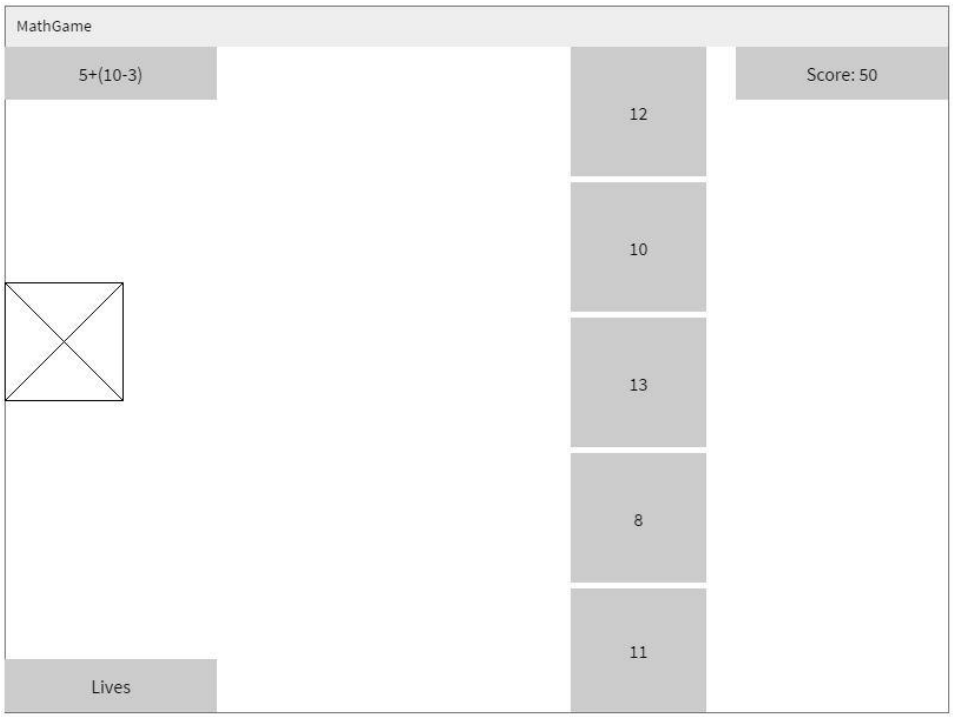

Figure 9 - game screen design (self authored)

As you can see in the upper part there are two text boxes. On the left side there is shown mathematical expression which player has to solve. On the other side there is shown play actual score. While moving down you can see player character which starts on the left corner of screen. Next thing is line of the blocks with numbers which splits up the game screen over the entire height. Lastly in the left bottom corner there are displayed heart sprites indicating the number of lives remaining.

4.4 Implementation

This part will depict how I went through the coding process. The first thing I did after creating the project was to create static class which stores most of important parameters for game logic. The reason for this was to distribute these mostly constant values through the entire application. I called it Method as it consists of variables which influence game play such like minimum and maximum length of mathematical expression, time which player has to answer, actual and score per question, remaining number of lives and more game variables.

4.4.1 Player character

Next thing was create class which will present a player character. However even before that I still needed to create another class – animation. In order not to player character look like unchanging sprite moving on the screen there need to be animation which will periodically change the sprite. Also because player need to contain four animations – for each direction – it is better idea to tear it away from player class and create specific class animation, maybe there could be another use for it somewhere else. To explain logic behind this class I will start at initialization where there needs to be added few slides in order to work. Each slide contains its sprite and also a given time for which it should be shown. Then core of animation is function called Update which does all logic behind – you can see the body of function on figure below.

```
public void Update(int timeElap)
{
    timeNow += timeElap;
    timeNow %= time;

    int tmp = 0;
    slideNow = -1;
    while (tmp <= timeNow)
    {
        slideNow++;
        tmp += slides[slideNow].Time;
    }
}
```

Basically it is timer which is periodically called and in each call is updating variable ‚timeNow‘. Then go through the list of slides and stops when the current elapsed time is

greater than slide time. In each step is incrementing variable ‘slideNow’ in order to point to current slide in list.

Now I can move over to player class which will consist of four animations for each direction plus one more for direction player is headed at given moment. Player class will also contain update method which besides updating actual animation will deal with keyboard control and positioning. Below you can see the whole update function.

```
public int Update(int elapsedTime, KeyboardState ks)
{
    animNow.Update(elapsedTime);
    int baseSpeed = Method.PlayerSpeed;
    if(keyCtrl) {
        if (ks.IsKeyDown(Keys.Left)) {
            speed.X = -baseSpeed;
            animNow = animLeft;
        }
        else if (ks.IsKeyDown(Keys.Right)) {
            speed.X = baseSpeed;
            animNow = animRight;
        }
        else speed.X = 0;

        if (ks.IsKeyDown(Keys.Down)) {
            speed.Y = baseSpeed;
            if (speed.X == 0) animNow = animDown;
        }
        else if (ks.IsKeyDown(Keys.Up)) {
            speed.Y = -baseSpeed;
            if (speed.X == 0) animNow = animUp;
        }
        else speed.Y = 0;
    }
    else {
        speed.X = baseSpeed;
        speed.Y = 0;
        animNow = animRight;
    }
    return UpdatePosition();
}
```

First thing after updating actual animation, is to check flag ‘keyCtrl’ - whether player should be controlled by keyboard. Reason for this approach is that after hitting correct answer game should assure player will reach the end of game screen and get into next level. If keyboard control is allowed there is check for pressed arrow keys and setting player character speed. In the other case there is firmly adjusted speed and animation in order to go in right

direction – to reach right corner of the screen. After speed is set it should also be projected into the position. For this reason there is called ‘UpdatePosition’ function which besides adding speed vector to the position one does also prevent player from stepping out of game window.

4.4.2 Answer blocks

Now that controllable player character is ready I can continue to blocks with answers. But here I will make use of inheritance and at first will create class for text boxes which will be useful later when I will be coding drawing of the game screen. Besides rectangle which stores position and size of the object class will also contain background sprite, and parameters for text – text itself, font and justification (alignment). Below you can see draw function which is except initialization only logic this class perform.

```
if(!initialised)
    return;

spriteBatch.Draw(background, pos, bgCol);

if (string.IsNullOrEmpty(text))
    return;
```

As you can see there are two check of possible application crash. First to check whether initialization is done since there wouldn’t be any settled position and color for the background. Next one is checking if it is necessary to draw the text. This check also prevent application crash in case there wouldn’t be initialized text of the object.

```
Vector2 txtPos = new Vector2();
if(just == eJustification.J_CENTER)
{
    txtPos.X = pos.X + pos.Width / 2 -
font.MeasureString(text).X / 2;
    txtPos.Y = pos.Y + pos.Height/2 -
font.MeasureString(text).Y/2;
}
else if(just == eJustification.J_LEFT)
{
    txtPos.X = pos.X + 1;
    txtPos.Y = pos.Y + pos.Height / 2 -
font.MeasureString(text).Y / 2;
}
else //if(just == eJustification.J_RIGHT)
{
    txtPos.X = pos.X + pos.Width - font.MeasureString(text).X;
    txtPos.Y = pos.Y + pos.Height / 2 -
font.MeasureString(text).Y / 2;
}
spriteBatch.DrawString(font, text, txtPos, fgCol);
```

ent on chosen alignment and then draws the text itself.

Now I can return to previous goal – to crate block class. Block class will be based on previously done text box although it will contain numeric value of text. Also there will be added random color generation for background. In order to player could distinguish blocks from each other there will be also check whether generated color is not already used by another block. In that case color generation will proceed again until it passes through this check.

4.4.3 Mathematical expression generator

At this moment I have prepared model of basic elements which user will see on the screen. However I am still far from completion, there still must be done collision check between each block and player character. But before that each of the blocks should have assigned its value. For this reason I need to find a way of generating mathematical expressions. I decided to create a class which will do this for me. This class needs to contain list of used operators, also some maximum/minimum number and length in order to be difficulty level dependent. Plus because it is going to generate numbers there is need of random – base class from .Net which provides random number generation. Main function of

this class is ‘GenerateExpression’ which includes two input parameters – expression length and flag whether brackets are allowed. During the function first is checked whether entering length is within the settled bounds. Then application will do for loop and into each even position it will add one from two possibilities – random generated number or brackets with sub-expression within.

```
rng = rnd.Next(101);
if(rng % 5 == 0 && allowBrackets) { // Add brackets
    if(length - i < minChars)
    {
        expr += this.GenerateNumber().ToString();
        continue;
    }

    int bLength = rnd.Next(minChars, length - i);

    if (bLength % 2 == 0)
        bLength -= 1;

    allowBrackets = false;
    i += bLength - 1;
    expr += "(" + GenerateExpression(bLength, false) + ")";
}
else { // Add random number
    expr += this.GenerateNumber().ToString();
}
```

tly set for testing, at least for now before settings screen and user parameters are added.

```
rng = rnd.Next(101);
for(int j = 0; j < operators.Count; j++) {
    if (rng % operators.Count == j) {
        expr += operators[j];
        break;
    }
}
```

4.4.4 Expression parser

Now that application has ready generation of mathematical expressions, I can display it to the user but one of the blocks must contain the correct answer. Besides initialization methods outside class is visible only one more function – Parse – which does internally calls function hidden inside the class. Functionality of this class works three steps – converting expression into the Reverse Polish Notation using Shunting yard algorithm and afterwards we do evaluation algorithm to calculate the result.

4.4.5 Game logic

Finally I have prepared every class for pure game functionality. What still remain is to associate them together with shared logic. But before I will go on to the code itself I would like to mention few important things. Main update and draw methods will be divided into parts dependent on game state – enum variable which tells us what is going on. Basically it is distinguishing game screens but there are few exceptions. Game itself is composed of three game states – running game, paused game and player collision with wrong block or limiting wall chasing him.

In order for game to be playable it needs to generate mathematical expression with blocks containing possible answers. On next figure you can see first part of method which provides this functionality.

```
if (expGen == null)
    expGen = new ExpressionGenerator();
if (parser == null)
    parser = new MathParser('.');

int len;
do
{
    len = rnd.Next(Method.ExprMinLen, Method.ExprMaxLen);
}
while (len % 2 == 0);
```

First thing this function does is safe check whether instances of expression generator and math parser classes are initialized. If not they will be initialized here. Next step is to generate

random length for expression. Since expression with even size would end in nonsense application need to assure length will be odd number.

```
mathExpr = expGen.GenerateExpression(len, true);

results = new double[Method.AnswersCount];
resCtrl = new List<double>();

PosCorrResult = rnd.Next(Method.AnswersCount);
double corrRes = parser.Parse(mathExpr, false);
```

Then I can make use of previously efforts and simply call functions for generating expression and calculating it right away. Now I have nearly everything but player need to have some possibilities to choose from. Since the correct result shouldn't be on the same place forever I will again make use of random and generate number which will be its index within array of all values.

```
resBlocks.Clear();
int bH = Method.WindowHeight / Method.AnswersCount;
for (int i = 0; i < Method.AnswersCount; i++)
{
    if (i == PosCorrResult)
        results[i] = corrRes;
    else
        results[i] = corrRes + GenerateFakeResult();

    Block newBlock = new Block(Method.BlocksPosX, 0 + (i * bH), 50, bH,
        blockBackground, blockFont, results[i]);
    resBlocks.Add(newBlock);
}
```

on the screen. In order to be easily changeable I put number of blocks to 'Method' class which I mentioned right at the beginning of implementation chapter.

Second most fundamental feature is to detect which answer player chosen. Basically there need to be check for collision between player and each of the blocks. Before I start to explain this functionality you can see the code itself on figure below.

```
foreach(Block b in resBlocks) {
    if(playerChar.CheckCollision(b.GetBlockRect()))
    {
        if(b.Value == results[PosCorrResult]) {
            playerChar.MoveToNextScreen();
            Method.IncreaseScore();
            scoreTR.SetText("Score: " + Method.Score);
        }
        else {
            g_State = GameState.GST_GameCollision;
            playerDeath.Reset();
            endTimerElapsed = 0;
        }
    }
}
```

Here application simply goes through entire list of blocks created before and check whether the current block collides with player character. For the check itself I made use of class Rectangle which comes from MonoGame framework and contains method which does detect whether two rectangles collide with each other. In case there is a collision application will continue with test to assure player chose correct one. If so player character is send to next screen by setting flag 'keyCtrl' in player class to false. This is done within function 'MoveToNextScreen'. Next player needs to be awarded for choosing correct answer by increasing score. And lastly program should update score text box in order to player could see his efforts. In other case – player chose wrong answer – I set current game state to collision where animation of explosion will be shown and then next expression will be generated. Needless to say this will cost player one of his precious lives.

As I mentioned before player has also time limit for each answer so I need to check when this time has passed and player did not choose any answer. I decided not to use simple timer but to make player chased by some kind of wall which will spread until it covers the entire screen. For this reason I need to add few parameters which will store maximum time of this limit. Because player is starting exactly at the left side of the screen wall needs to be hidden for a while to provide player some time to move. I decided to split this time into two variables – time for which wall is hidden and time which takes the wall to spread towards

blocks. And instead of checking time I will check for collision of player with wall chasing him – after passing first part of timer.

4.4.6 High Scores storing

Before I will start doing user interface of the application there is still more to do first. In order to display the really best scores when player goes to given screen I need to make class which will store them for me. However storing won't be only function for this class since application should be keeping old scores even from different time when it was running. For this purpose I will make save function which will be called each time top ten scores will change. High scores will be saved into .dat file into application folders. Now I must do a function for loading them to the application memory. This function will be called once per application run – at the very start during the initialization phase. When player will lost his last life current achieved score will be send to Highscores class in order to check whether it is worthy of keeping. If it is flag "achievedHS" which will allow additional animation of fireworks to be shown will be set.

4.4.7 Game interface and settings

When main part of game – the game itself – is ready I can move to user interface in order to provide secondary features of the application such as settings and high scores screen. First I will focus on creating a menu which will provide basic navigation for the player through the game. Menu will be done via buttons controlled by mouse. Class for button will contain three Texture2D variables – sprite, sprite selected and sprite now – and position on the screen. Main logic for buttons will be done in game update method. This part of method you can see on the following figure.

```
Vector2 mousePos = new Vector2(Mouse.GetState().X,
Mouse.GetState().Y);
bool mousePressed = (Mouse.GetState().LeftButton ==
ButtonState.Pressed);
if (nGameBtn.MouseOver(mousePos) && mousePressed)
{
    if (!Method.SettingsFault)
    {
        g_State = GameState.GST_Game;
        InitNewGame();
    }
}
if (scoresBtn.MouseOver(mousePos) && mousePressed)
{
    g_State = GameState.GST_HighScores;
    InitScoreTextRects();
}
if (settingsBtn.MouseOver(mousePos) && mousePressed)
    g_State = GameState.GST_Settings;
if (endBtn.MouseOver(mousePos) && mousePressed)
    g_State = GameState.GST_End;
```

o variables. Then I can move to testing when player clicks any of buttons. In order to update sprite of the button first in each condition will be tested whether mouse is hovering over the button since in function "MouseOver" there is updated sprite now within button class. Dependent on which button was clicked functionality will be performed. In case of new game button, game state will be changed and initialization of new game is done by generating mathematical expression and resetting parameters – number of lives and current score. When going into high scores screen there will be called function for updating text boxes in order to contain the current top scores.

For next screen – settings of game parameters – I need to create few objects before I can start with screen itself. First I will create class for storing set of user parameters which can be then accessed from anywhere in code. For this reason I will make whole class static so I will call its function by the class itself. Even before that I need to make a structure for user parameter. It should contain some kind of identification – in my case I chose text – also description what it deals with, next important variables are type of parameter – enum, text, whole or decimal number. For each type there will be some variables to restrict its value and of course value itself. Since I will use only enum parameters I will start by explaining these. For enum parameters there will be text list storing text values which can be chosen. For pointing on current value there will be used integer index with also minimum and maximum value. Like

for high scores these parameters will also be saved and load from .dat file in order to memorize user settings.

Next thing before I can start coding the screen is to create objects which will provide functionality for user to change current settings. For this purpose I took text box as a base class and extended it into class I called “EnumTextRect” which will contain additional logic for changing setting via clicking on text box itself. This class will need to additionally contain user parameter which will be changing. Since the whole logic will be inside the class I decided to insert all these instances into list for better control. In game update function I will simply for each of text boxes call update method which you can see on following figure.

```
public void Update(MouseState ms, int elapsedTime)
{
    _clicked = ms.LeftButton == ButtonState.Pressed;
    if (_oldClicked && !_clicked)
    {
        Vector2 m_pos = ms.Position.ToVector2();
        if (m_pos.X >= base.pos.X && m_pos.X <= base.pos.X +
base.pos.Width
            && m_pos.Y >= base.pos.Y && m_pos.Y <= base.pos.Y +
base.pos.Height)
        {
            _prm.SetValue(++_prm.value.i);
            base.SetText(_prm.hint + ": " +
(string)_prm.GetValue());
        }
    }
    _oldClicked = _clicked;
}
```

single call of update since there is delay in milliseconds between each call. For this purpose I am checking the edge of click – when player release the mouse button. To be able do this I am keeping last state of mouse which must be pressed but the current state should be released in order this event to occur. When event is fired I simply increase value of parameter index to point on next text. Afterwards I can update text in the box in order to stay up-to-date.

At this moment settings will consist of difficulty level and allowing four of basic math operators – plus, minus, division and multiplication. These parameters will be read when leaving settings screen and expression generator will be update to project possible changes.

5 Results and Discussion

The application is completed and is fulfilling the main objective to provide practicing of basic mathematical calculations. On next figure you can see final appearance of game screen which strongly resembles the original design.

Figure 10 - final game screen (self authored)

5.1 Possible ways of future development

During the implementation process I came across of few ideas which could enhance my final solution to even better state.

5.1.1 Optimization

There is still place for improving already existing features. For example it could be done optimization of mathematical expression generation to prevent division by zero which could occur in some special case. Another aspect which could be improved is graphical part – improving background images for non-game screens.

5.1.2 Generalize the theme of orientation

I could upgrade my game with modules for practicing not only calculation of mathematical expression but any subject. Although current concept of generating questions wouldn't work in this case, there is possibility to use database for storing sets of questions and just load set dependent on current theme. Besides sets of questions database would need to contain answers for them – with ability to distinguish correct and wrong one. Also kind of indexing sets of question in order to separate different themes when loading questions.

5.1.3 Making real-time multiplayer

Another possibility of upgrading final solution is adding multiplayer mode where players could engage themselves in real-timed challenge in order to find out who is better. There could be two different modes – two players on same computer, each player on his computer via LAN.

6 Conclusion

To recapitulate let us return to Objectives chapter. The main goal was stated as designing and implementing educational game. Designing part was done during chapter 4.2 Game specification where I described the features and main logic of the game. Next part of design was mentioned during chapter 4.3 Interface design where I depicted layout of graphical user interface for each of screens and some of their functionality. Second part of main goal – implementing – which steps are described in chapter 4.4 Implementation while each subchapter is kind of step to be done. For the sub-goals I briefly explained during literature review fundamentals of .Net and MonoGame frameworks which are used for implementation of final solution. Then was described theory behind calculating algorithm inside the application in order to check correct answers.

7 References

Čápka, D. (n.d.). *Úvod do C# a .NET frameworku*. Retrieved from IT network: http://www.itnetwork.cz/csharp/zaklady/c-sharp-tutorial-uvod-do-jazyka-a-dot-net-framework

Fleischauer, M. (2015). *Cross Platform Game Development with MonoGame.*

Howland, G. (n.d.). *How do I make games? A Path to Game Development*. Retrieved from gamedev.net: https://www.gamedev.net/resources/_/technical/game-programming/how-do-i-make-games-a-path-to-game-development-r892

Microsoft. (n.d.). *Microsoft.Xna.Framework.Graphics Namespace*. Retrieved from Microsoft Developer Network: https://msdn.microsoft.com/en-us/library/microsoft.xna.framework.graphics.aspx

Spilman, T. (n.d.). Retrieved from MonoGame: http://www.monogame.net/about/

Tan, V. (n.d.). *Reverse Polish Notation In C#*. Retrieved from dreamincode: http://www.dreamincode.net/forums/topic/35320-reverse-polish-notation-in-c%23/

Whitaker, R. (n.d.). *Monogame - Introduction To 2D Graphics*. Retrieved from RB Whitaker's Wiki: http://rbwhitaker.wikidot.com/monogame-introduction-to-2d-graphics

8 Appendix

Source codes of the game itself are present on the CD given with this thesis.