Ministry of Education and Science of Ukraine

Odessa National Academy of Food Technologies

International Competition of Student Scientific Works

BLACK SEA SCIENCE 2021

Information Technology, Automation and Robotics

Proceedings

Odessa, ONAFT 2021

UDC 004.01/08

Editorial board:

Black Sea Science 2021: Proceedings of the International Competition of Student Scientific Works. Information Technology, Automation and Robotics. / Odessa National Academy of Food Technologies; B.Yegorov, M. Mardar, S.Kotlyk (editors-in-chief.) [*et al.*]. – Odessa: ONAFT, 2021. – 526 p.

These materials of International Competition of Student Scientific Works «Black Sea Science 2021» contain the works of the contest participants in the section «Information technologies, automation and robotics» (not winners).

The author of the work is responsible for the accuracy of the information.

Organizing committee:

Prof. Bogdan Iegorov, D.Sc., Rector of Odessa National Academy of Food Technologies, Head of the Committee

Prof. Maryna Mardar, D.Sc., Vice-Rector for Scientific and Pedagogical Work and International Relations of Odessa National Academy of Food Technologies, Deputy Head of the Committee

Prof. Stefan Dragoev, D.Sc., Vice-Rector for Scientific Work and Business Partnerships of University of Food Technologies (Bulgaria)

Prof. Baurzhan Nurakhmetov, D.Sc., First Vice-Rector of Almaty Technological University (Kazakhstan)

Prof. Mircea Bernic, Dr. habil., Vice-Rector for Scientific Work of Technical University of Moldova (Moldova)

Prof. Jacek Wrobel, Dr. habil., Rector of West Pomeranian University of Technology (Poland)

Prof. Michael Zinigrad, D.Sc., Rector of Ariel University (Israel)

Dr. Mei Lehe, Ph.D., Vice-President of Ningbo Institute of Technology, Zhejiang University (China)

Prof. Plamen Kangalov, Ph.D., Vice-Rector for Academic Affairs of "Angel Kanchev" University of Ruse (Bulgaria)

Dr. Alexander Sychev, Ph.D., Assoc. Professor of Sukhoi State Technical University of Gomel (Belarus)

Dr. Hanna Lilishentseva, Ph.D., Assoc. Professor, Head of the Department of Merchandise of Foodstuff of Belarus State Economic University (Belarus)

Prof. Heinz Leuenberger, Ph.D., Professor of the Institute of Ecopreneurship of University of Applied Sciences and Arts (Switzerland)

Prof. Edward Pospiech, Dr. habil., Professor of the Institute of Meat Technology of Poznan University of Life Sciences (Poland)

Prof. Lali Elanidze, Ph.D., Professor of the Faculty of Agrarian Sciences of Iakob Gogebashvili Telavi State University (Georgia)

Dr. V. Kozhevnikova, Ph.D., Senior Lecturer of the Department of Hotel and Catering Business of Odessa National Academy of Food Technologies, Secretary of the Committee

The jury for the section «Information technologies, automation and robotics»

Head of the jury:

Sergii Kotlyk – Ph.D., Associate Professor, Director of the P.M. Platonov Educational-Scientific Institute of Computer Systems and Technologies "Industry 4.0" of Odessa National Academy of Food Technologies (Ukraine)

Members of the jury:

Piotr Artiemjew - Dr hab., Associate Professor in Decision Systems of the Faculty of Mathematics and Computer Science,University of Warmia and Mazury in Olsztyn (Poland)

Francisco Antonio Augusto – Dr., International Relations Manager of Higher Institute of Information and Communication Technologies (Angola)

Andrey Kuprijanov – Ph.D., Associate Professor of the Department of Software for Computers and Automated Systems of Belarusian National Technical University (Belarus)

Simon Milbert – Vice-President of Xtra Information Management, Inc. (USA)

Ivan Palov – D.Sc., Professor of University of Ruse "Angel Kanchev" (Bulgaria)

Degla Gérard Hugues – Communications and Training Manager of "MAPCOM solutions informatiques" company group (Benin)

Nugzar Kereselidze - Academic Doctor of Informatics (Computer Science), Associate Professor of the Department of Natural Sciences, Mathematics, Technology and Pharmacy, Sukhumi State University (Georgia)

Etibar Seyidzade - Associate Professor of the Department of Computer and Information Technolohies, Baku Engineering University (Azerbaijan)

Vladimir Golenkov, D.Sc., Professor of the Department of Intelligent Information Technologies, Belarusian State University of Informatics and Radio Electronics (Belarus)

Zhanar Omirbekova - Ph.D., Associate Professor of the Department of Automation and Management, Satbayev University (Kazakhstan)

Ivan Palov - D.Sc., Professor of the Department of Power Supply and Electrical Equipment, University of Ruse "Angel Kanchev" (Bulgaria)

Siarhei Palavenia - Ph.D., Associate Professor, Head of the Department of Telecommunication Systems, Belarusian State Academy of Communications (Belarus)

Alexander Goloskokov - Ph.D., Professor of the Department of Software Engineering and Information Technology Management, National Technical University "Kharkiv Polytechnic Institute" (Ukraine)

Peter Nikolyuk - D.Sc., Professor of the Department of Computer Technology, Vasyl Stus Donetsk National University (Ukraine)

Vladimir Palagin - D.Sc., Professor, Head of the Department of Radio Engineering, Telecommunications and Robotics Systems, Cherkasy State Technological University (Ukraine)

Viktor Khobin – D.Sc., Professor, Head of the Department of Technological Processes Automation and Robotic Systems of Odessa National Academy of Food Technologies (Ukraine)

Valeriy Plotnikov – D.Sc., Professor, Head of the Department of Information Technology and Cybersecurity of Odessa National Academy of Food Technologies (Ukraine)

Sergii Artemenko – D.Sc., Professor, Head of the Department of Computer Engineering of Odessa National Academy of Food Technologies (Ukraine)

Fedir Trishyn - Ph.D., Associate Professor, Vice-Rector on Scientific and Educational Work, Odessa National Academy of Food Technologies (Ukraine)

Valerii Levinskyi – Ph.D., Associate Professor of the Department of Technological Processes Automation and Robotic Systems of Odessa National Academy of Food Technologies (Ukraine)

Viktor Yehorov – Ph.D., Supervisor of the Laboratory of Mechatronics and Robotics of Odessa National Academy of Food Technologies (Ukraine)

Pavlo Lomovtsev – Ph.D., Associate Professor of the Department of Information Technology and Cybersecurity of Odessa National Academy of Food Technologies (Ukraine)

Yurii Kornienko – Ph.D., Associate Professor of the Department of Information Technology and Cybersecurity of Odessa National Academy of Food Technologies (Ukraine)

Serhii Shestopalov – Ph.D., Associate Professor of the Department of Computer Engineering of Odessa National Academy of Food Technologies (Ukraine)

Anatoly Galiulin - Ph.D., Associate Professor, Acting Head of the Department of Electromechanics and Mechatronics, Odessa National Academy of Food Technologies (Ukraine)

Secretary of the jury:

Oksana Sokolova – Senior Lecturer of the Department of Information Technology and Cybersecurity of Odessa National Academy of Food Technologies (Ukraine)

aperture antenna systems that will increase the performance of space communication systems.

This device will be designed to perform rural work related to measuring the parameters of wide-aperture antennas and research work related to their development.

VI. REFERENCES

1. Miriam McNabb (2016, April 08). *Changing Forecasts: The Drone Industry Surprise*. DroneLife. https://dronelife.com/2016/04/08/comparing-drone-industry-forecasts/

2. Wikipedia (2021, January 27). *Unmanned Aerial Vehicle*. https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle

3. Ordodi Marton (1984). *Hang gliding*. Fly-history. http://fly-history.ru/books/item/f00/s00/z0000012/index.shtml

4. Richard Vaughn (2013, December 02). *The difference between Cartesian, Six-Axis, and SCARA robots*. MachineDesign. https://www.machinedesign.com/mechanical-motion-systems/article/21831692/the-difference-between-cartesian-sixaxis-and-scara-robots

5. Wikipedia (2019, November 07). *NanoPi Neo Core2*. https://wiki.friendlyarm.com/wiki/index.php/NanoPi_NEO_Core2

DEVELOPMENT OF ELECTRONIC APPLICATION FOR RENDERING OF BEZIER CURVES

Author: ***Andrii Kurhanskyi***
Advisor: ***Nadiia Olefirenko***
H. S. Skovoroda Kharkiv National Pedagogic University (Ukraine)

Abstract. *The work is devoted to the development of a program for rendering one of the types of parametric curves - Bezier curves. These curves have already become an important part of computer design. In the course of the work, the author developed an electronic application using the C ++ / CLI programming language. This app helps to demonstrate the process of rendering a Bezier curve. The paper shows the historical and mathematical aspects of the topic, talks about writing code and using Bezier curves.*

Key words. *Bezier curves, de Casteljau algorithm, development, electronic application, C++/CLI, Windows Forms, GUI.*

I. INTODUCTION

Bezier curves have influenced computer graphics a lot. They are in almost every computer graphics processor. But none of them can show us the process of rendering Bezier curves, which is a very interesting process, especially for students of a secondary school.

The purpose of work is to develop electronic application for demonstrating the process of rendering Bezier curves. For this purpose, we offer to use the geometric

interpretation of recursive de Casteljau algorithm in algorithmic part of work, the Visual C++/CLI 17 WinForms on platform .NET Framework v.4.7.2. with SDK v.10.0 as hard tools for development of the application and Microsoft Visual Studio 2019 Community as IDE.

II. LITERATURE ANALYSIS

2.1 History of creating of Bezier curves

In order to solve CAD/CAM mathematical problems, many solutions have been offered, each adapted to specific matters.

Around 1960, designers of stamped parts, that is, car-body panels, used french curves and sweeps, but in fact, the final standard was the "master model," the shape of which, for many valid reasons, could not coincide with the curves traced on the drawing board. This resulted in discussions, arguments, haggling, retouches, expenses, and delay. Obviously, no significant improvement could be expected so long as a method was not devised that could prove an accurate, complete, and undisputable definition of freeform shapes.

Computing and numerical control (NC), at that time, had made great progress, and it was certain that only numbers, transmitted from drawing office to tool drawing office, manufacturing, patternshop, and inspection could provide an answer; of course, drawings would remain necessary, but they would only be explanatory, their accuracy having no importance, and numbers being the only and final definition.

Certainly, no system could be devised without the help of mathematics—yet designers, who would be in charge of operating it, had a good knowledge of geometry, especially descriptive geometry, but no basic training in algebra or analysis.

However, in the 1960s a mathematician and engineer named Pierre Bezier changed everything with his newly developed CAGD tool called UNISURF. This new software allowed designers to draw smooth looking curves on a computer screen, and used less physical storage space for design materials.

In France, at that time, very little was known about the work performed in the American aircraft industry; the papers from James Ferguson were not much displayed before 1964; Citroen was secretive about the results obtained by Paul de Casteljau, and the famous technical report MAC-TR-41 (by S. A. Coons) did not appear before 1967; The works of W. Gordon and R. Riesenfeld were printed in 1974.

At the beginning, the idea of UNISURF was oriented toward geometry rather than analysis, but with the idea that every datum should be exclusively expressed by numbers.

2.2 Usage of Bezier curves

Bezier curves are widely used in computer graphics to model smooth curves, in animation to create two-dimensional and three-dimensional paths for moving objects and in fonts for drawing curved shapes.

Fig. 1. Letter “e” made with Bezier curve

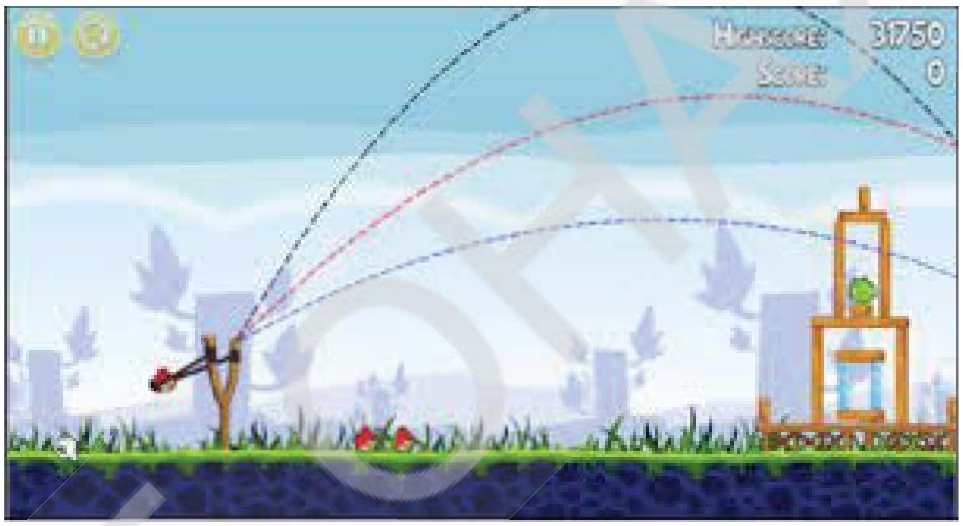

Fig. 2. In game “Angry birds” path of fly is Bezier curve

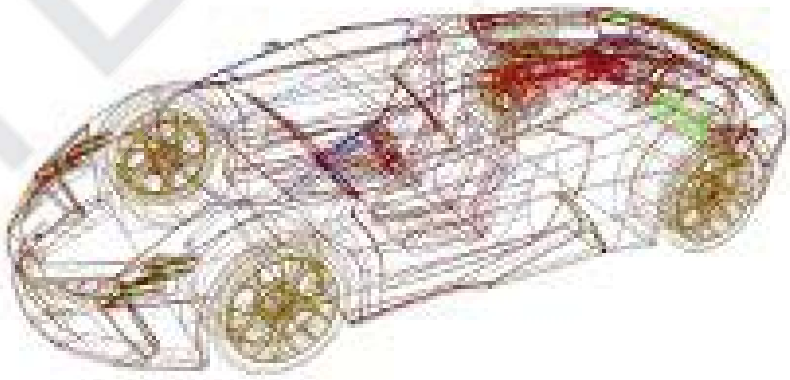

Fig. 3. Model of a sport car, designed with Bezier curves

2.3. Mathematical basics of building of Bezier curves

Bezier began with the idea that any point p(u) on a curve segment should be given by an equation such as the following:

$$p(u) = \sum_{i=0}^{n} p_i f_i(u) \qquad (1)$$

where $0 \leq u \leq 1$, and the vectors p_i are the control points.

The n + 1 functions, that is the $f_i(u)$,must produce a curve that has certain well-defined characteristics. Here are some of the most important ones:

1. The curve must start on the first control point, p_0, and end on the last, p_n. Mathematically, we say that the functions must interpolate these two points.

2. The curve must be tangent to the line given by $p_i - p_0$ at p_0 and to $p_n - p_{n-1}$ at p_n.

3. The functions $f_i(u)$ must be symmetric with respect to u and (1 - u). This lets us reverse the sequence of control points without changing the shape of the curve.

Other characteristics can be found in more advanced works on this subject.

A family of functions called Bernstein polynomials satisfies these requirements. They are the basis functions of the Bezier curve. (Other curves, such as the NURBS curves, use different, but related, basis functions.) We rewrite Equation 1 using them, so that

$$p(u) = \sum_{i=0}^{n} p_i B_{i,n}(u) \qquad (2)$$

where the basis functions are

$$B_{i,n}(u) = \binom{n}{k} u^i (1-u)^{n-i} \qquad (3)$$

III. OBJECT, SUBJECT AND METHODS OF RESEARCH

Object: Modeling of parametric curves.

Subject: Creating the program for rendering Bezier curves.

Methods of research: In this work were used methods of analysis of literature in this topic, creating of personal electronic application to have dynamic process of rendering Bezier curves for any amount points build curve of needed power, even if hull is two-dimensional projection of three-dimensional geometric solid.

IV. RESULTS

4.1. Development of application's graphical user interface

First of all, GUI (Graphical User Interface) should have picture box to opportunity enter points and display process of rendering of Bezier curves, also interface should have two buttons: "Build" and "Clear". When someone press button "Build" after entering random amount of points program has to demonstrate process of building of Bezier curves with given parameters, this button erases additional lines which should be changed for every frame. And button "Clear" has to complete options which have shown in its name – clear workspace, it can be released with filling workspace with white color.

However, it is possible to add another one button – "Build with lines", that is similar to button "Build" but differ from first button this one does not erase additional

lines, thanks to this process of rendering goes easily than such process in case of pressing button “Build”.

All buttons are included in control panel. This component of interface also contains label for demonstrating amount of points which were entered in picture box with pressing of left mouse button.

Fig. 4. View of program’s GUI

Figure 1 shows graphical user interface when user has just opened program. All elements that included in window. It is possible to resize window and make in full screen size. Figure 2 demonstrates that all elements are extendable to fit of the window size.

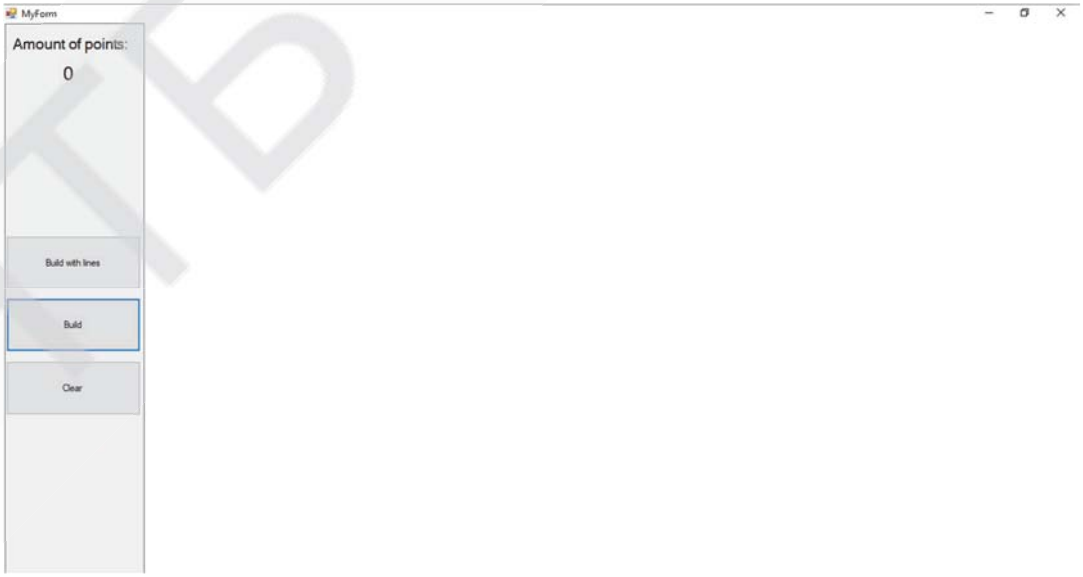

Fig. 5. Fullscreen view of program’s GUI

4.2. Development of program structure

In modern programming practice, it's desirable to use object-oriented style of coding. It helps to protect data from changing outside of class body.

Main file of project initializes window (form), runs it, contains including of headers and commands to use system namespaces.

Source code of MyForm.h contains namespace which has class to initialize all components of form, also this class defines private void methods to rule the behavior of users' actions. But hole namespace does not define constants for window settings, it makes in header which goes with IDE - windows.h.

Class includes such methods as "BuildLine1", "Newcord", "BuildLine2", "FindBezier", "DrawBezier", "DrawEllipses", "workofbuttonbuild", "workofbuttonbuildwithlines", "button1_Click", "button3_Click" and "pictureBox1_MouseDown".

4.3. Development of basic program's class methods

```
private: void BuildLine1() {
                Graphics^ g = pictureBox1->CreateGraphics();
                g->SmoothingMode =
System::Drawing::Drawing2D::SmoothingMode::HighQuality;
                Pen^ pen2 = gcnew Pen(Brushes::Black);
                pen2->Width = 2.0f;
                for (int i = 1; i < points; ++i) {
                        Point a, b;
                        a.X = cordx[i - 1];
                        a.Y = cordy[i - 1];
                        b.X = cordx[i];
                        b.Y = cordy[i];
                        g->DrawLine(pen2, a, b);
                }
        }
```

Listing 1. Source code for method "BuildLine1"

In listing 1 we create private void class method with name "BuildLine1". It has no parameters. Next line creates and initializes descriptor for pictureBox1. After it goes setting of high-quality smoothing mode. Next, we create and initialize pen for drawing and set black color to this one. Set width of pen 2 pixels. And in loop for define structures as point and initialize them, draw line between two points and repeat it for all points (for point we take this one and previous).

```
private: void Newcord(int j, double t) {
                double lambda = t / (1 - t);
                for (int i = 0; i < points-1; ++i) {
                        ncordx[j][i] = (ncordx[j][i] + lambda * ncordx[j-1][i + 1])/(1+lambda);
                        ncordy[j][i] = (ncordy[j][i] + lambda * ncordy[j-1][i + 1])/(1+lambda);
                }
        }
```

Listing 2. Source code for method “Newcord”

In listing 2 we create private void class method with name “Newcord”. It has two parameters, number of current iteration and parameter for Bezier curve. Initialize variable lambda as double type for dividing line. Next, divide all lines in given relation.

```
private: void BuildLine2(int j) {
                Graphics^ g = pictureBox1->CreateGraphics();
                g->SmoothingMode =
System::Drawing::Drawing2D::SmoothingMode::HighQuality;
                Pen^ pen2 = gcnew Pen(Brushes::Green);
                pen2->Width = 2.0f;
                Point a, b;
                for (int i = 1; i <= points - j; ++i) {
                        a.X = ncordx[j][i - 1];
                        a.Y = ncordy[j][i - 1];
                        b.X = ncordx[j][i];
                        b.Y = ncordy[j][i];
                        g->DrawLine(pen2, a, b);
                }
        }
```

Listing 3. Source code for method “BuildLine2”

In listing 3 we create private void class method with name “BuildLine2”. It has one parameter, number of current iteration. Next line creates and initializes descriptor for pictureBox1. After it goes setting of high-quality smoothing mode. Next, we create and initialize pen for drawing and set black color to this one. Set width of pen 2 pixels. And in loop for define structures as point and initialize them, draw additional line between two points for this iteration and repeat it for all points (for point we take this one and previous).

```
private: void FindBezier(double t) {
                double lambda = t / (1 - t);
                beziercordx.push_back((ncordx[points-1][0] + lambda * ncordx[points-1][1]) / (1
+ lambda));
                beziercordy.push_back((ncordy[points-1][0] + lambda * ncordy[points-1][1]) / (1
+ lambda));
        }
```

Listing 4. Source code for method “FindBezier”

In listing 4 we create private void class method with name “FindBezier”. It has one parameter, parameter for Bezier curve. Initialize variable lambda as double type for dividing line. Next, divide all additional lines in given relation to find coordinates of point in Bezier curve.

```
private: void DrawBezier() {
                Graphics^ g = pictureBox1->CreateGraphics();
                g->SmoothingMode =
System::Drawing::Drawing2D::SmoothingMode::HighQuality;
                Pen^ pen1 = gcnew Pen(Brushes::Red, 4);
                Point a, b;
                for (int i = 1; i < beziercordx.size(); ++i) {
                        a.X = beziercordx[i - 1];
                        a.Y = beziercordy[i - 1];
                        b.X = beziercordx[i];
                        b.Y = beziercordy[i];
                        g->DrawLine(pen1, a, b);
                }
        }
```

Listing 5. Source code for method “DrawBezier”

In listing 5 we create private void class method with name “DrawBezier”. It has no parameters. Next line creates and initializes descriptor for pictureBox1. After it goes setting of high-quality smoothing mode. Next, we create and initialize pen for drawing and set black color to this one. Set width of pen 2 pixels. And in loop for define structures as point and initialize them, draw additional line between two points in Bezier curve for this iteration and repeat it for all points (for point we take this one and previous) to draw part or hole Bezier curve.

```
private: void DrawEllipses() {
                Graphics^ g = pictureBox1->CreateGraphics();
                g->SmoothingMode =
System::Drawing::Drawing2D::SmoothingMode::HighQuality;
                Pen^ pen1 = gcnew Pen(Brushes::Black);
                pen1->Width = 5.0f;
                for (int i = 0; i < points; ++i) {
                        g->DrawEllipse(pen1, cordx[i], cordy[i], 5, 5);
                }
        }
```

Listing 6. Source code for method “DrawEllipses”

In listing 6 we create private void class method with name “DrawEllipses”. It has no parameters. Next line creates and initializes descriptor for pictureBox1. After it goes setting of high-quality smoothing mode. Next, we create and initialize pen for drawing and set black color to this one. Set width of pen 5 pixels. And draws all points entered by user for next iterations.

```
private: void workofbuttonbuild() {
		try {
			if (points < 2) {
				String^ s = "Exception 1: Not enough points. Put at least 2 points";
				throw s;
			}
			if (points > 20) {
				String^ s = "Exception 2: Too much points. Put less points";
				throw s;
			}
			Graphics^ g = pictureBox1->CreateGraphics();
			g->SmoothingMode =
System::Drawing::Drawing2D::SmoothingMode::HighQuality;
			g->Clear(Color::White);
			DrawEllipses();
			BuildLine1();
			ncordx.resize(points);
			ncordy.resize(points);
			for (int i = points; i > 1; --i) {
				ncordx[points - i].resize(i);
				ncordy[points - i].resize(i);
			}
			ncordx[0] = cordx;
			ncordy[0] = cordy;
			ncordx[1] = cordx;
			ncordy[1] = cordy;
			beziercordx.clear();
			beziercordy.clear();
			for (double i = 0.0f; i < 1; i += 0.004) {
				for (int j = 2; j < points; ++j) {
					ncordx[j] = ncordx[j - 1];
					ncordy[j] = ncordy[j - 1];
					Newcord(j, i);
					BuildLine2(j);
				}
				FindBezier(i);
				DrawBezier();
				_sleep(60);
				g->Clear(Color::White);
				DrawEllipses();
				BuildLine1();
			}
			DrawBezier();
		}
		catch (String^ s) {
			MessageBox::Show(s);
		}
	}
```

Listing 7. Source code for method “workofbuttonbuild”

In listing 7 we create private void class method with name "workofbuttonbuild". It has no parameters. We use try – catch construction. In try part we check amount of entered points and throw exceptions to catch part. Next line creates and initializes descriptor for pictureBox1. After it goes setting of high-quality smoothing mode. Next, fill workspace with white color. Call methods "DrawEllipses" and "BuildLine1". Resize vectors (dynamic arrays) and two-dimensional vectors. Initialize first and second elements of two-dimensional vectors. Clear vector of Bezier curve coordinates. Two loops "for" change elements of vector and call methods "Newcord" and "BuildLine2". Next, in first loop "for" we call methods "FindBezier", "DrawBezier", wait for 60 milliseconds, clear workspace and call methods "DrawEllipses" and "BuildLine1". After all we call method "DrawBezier". In catch part we take exception message as pointer to string type and show messagebox with the text of exception.

Method "workofbuttonbuildwithlines" is similar to previous, but it does not clear workspace so all additional lines stay on it.

```
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
            th1 = gcnew System::Threading::Thread(gcnew
System::Threading::ThreadStart(this, &MyForm::workofbuttonbuild));
            th1->Start();
            thc1 = true;
        }
```

Listing 8. Source code for method "button1_Click"

In listing 8 we create private void class method with name "button1_Click". It has standard parameters. We create a new thread to call method "workofbuttonbuild", next it starts and logical variable is set value true.

Method "button2_Click" work at the same way.

```
private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e) {
        Graphics^ g = pictureBox1->CreateGraphics();
        g->SmoothingMode = System::Drawing::Drawing2D::SmoothingMode::HighQuality;
        g->Clear(Color::White);
        points = 0;
        label1->Text = points.ToString();
        cordx.clear();
        cordy.clear();
        ncordx.clear();
        ncordy.clear();
        if (thc1) {
            th1->Abort();
            thc1 = false;
        }
        if (thc2) {
            th2->Abort();
            thc2 = false;
        }
}
```

Listing 9. Source code for method "button3_Click"

In listing 9 we create private void class method with name “button3_Click”. It has standard parameters. Next line creates and initializes descriptor for pictureBox1. After goes setting of high-quality smoothing mode. Next, fill workspace with white color, set zero to variable points, change text in label and clear all vectors (dynamic arrays). After that, abort started thread and set false to logical variable.

```
private: System::Void pictureBox1_MouseDown(System::Object^ sender,
System::Windows::Forms::MouseEventArgs^ e) {
        cordx.push_back(e->X);
        cordy.push_back(e->Y);
        Graphics^ g = pictureBox1->CreateGraphics();;
        g->SmoothingMode = System::Drawing::Drawing2D::SmoothingMode::HighQuality;
        Pen^ pen1 = gcnew Pen(Brushes::Black);
        pen1->Width = 5.0f;
        g->DrawEllipse(pen1, e->X, e->Y, 5, 5);
        points++;
        label1->Text = points.ToString();
        BuildLine1();
}
```

Listing 10. Source code for method “pictureBox1_MouseDown”

In listing 10 we create private void class method with name “BuildLine1”. It has standard parameters. Memorize coordinates of entered points. Next line creates and initializes descriptor for pictureBox1. After goes setting of high-quality smoothing mode. Next, we create and initialize pen for drawing and set black color to this one. Set width of pen 5 pixels. And draws point entered by user. Increment variable points, change label and call method “BuildLine1”.

4.4. Results of application’s work

All pictures in this chapter are screenshots, which were made during it was working. First picture in every row is broken line for building Bezier curve, second – is process of rendering and third - is result.

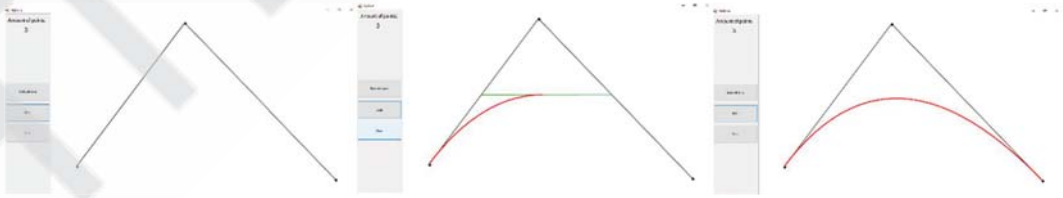

Fig. 6-8. Bezier curve of second power with convex hull

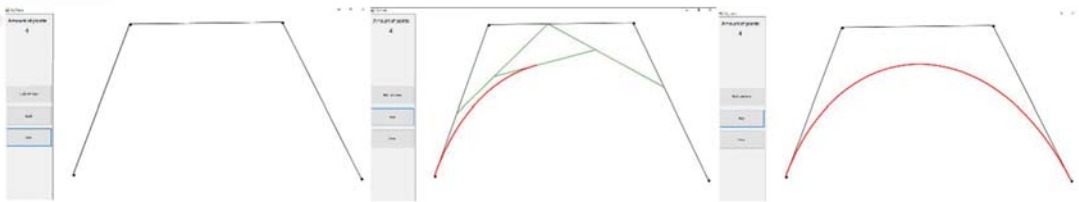

Fig. 9-11. Bezier curve of third power with convex hull

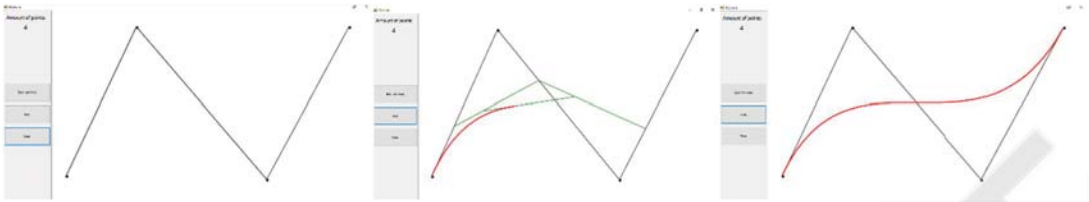

Fig. 12-14. Bezier curve of third power with non-convex hull

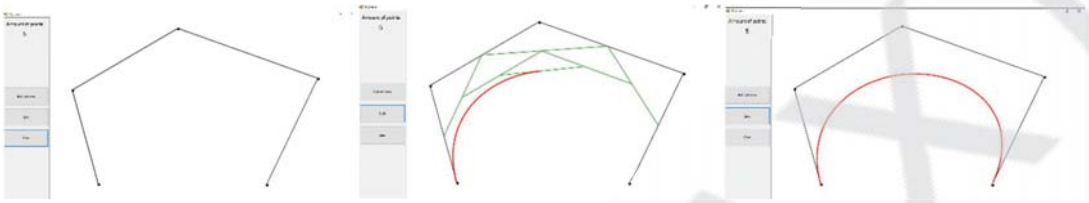

Fig. 15-17. Bezier curve of fourth power with convex hull

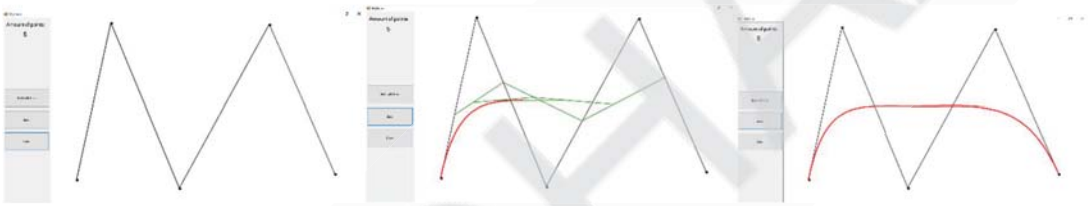

Fig. 18-20. Bezier curve of fourth power with non-convex hull

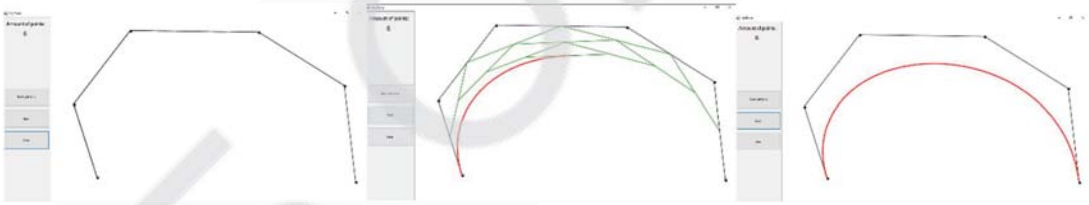

Fig. 21-23. Bezier curve of fifth power with convex hull

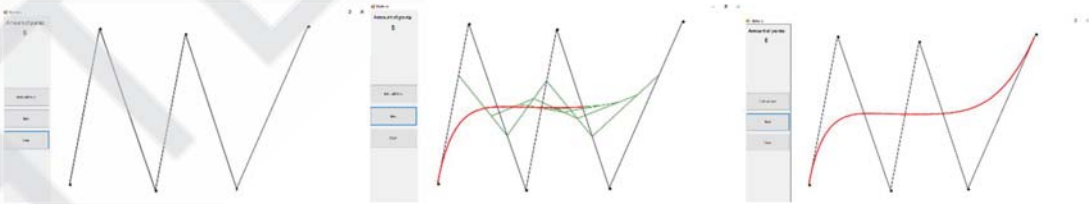

Fig. 24-26. Bezier curve of fifth power with non-convex hull

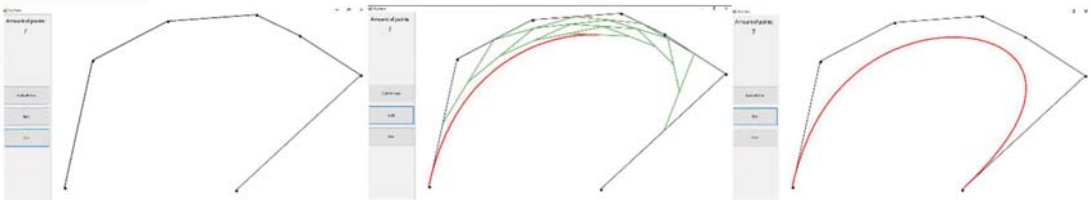

Fig. 27-29. Bezier curve of sixth power with convex hull

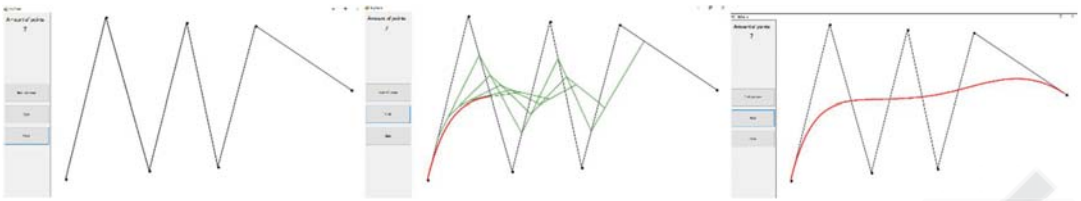

Fig. 30-32. Bezier curve of sixth power with non-convex hull

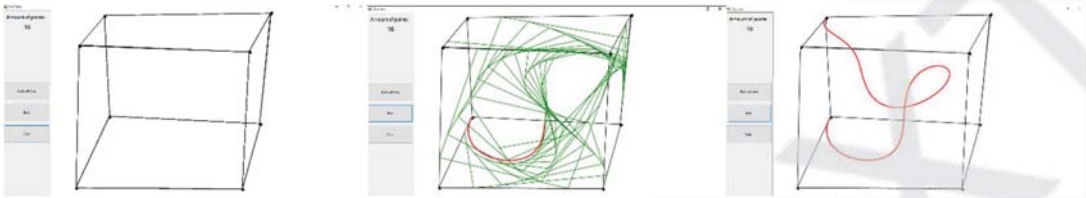

Fig. 33-35. Bezier curve built in hull of 2D projection of 3D cube

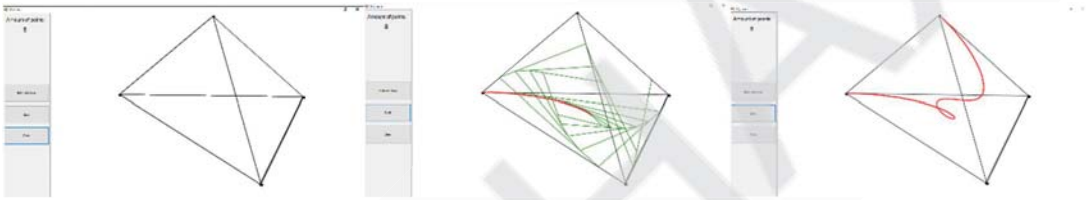

Fig. 36-38. Bezier curve built in hull of 2D projection of 3D pyramid

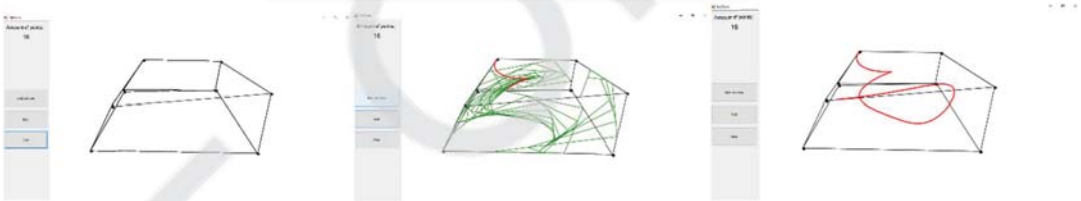

Fig. 39-41. Bezier curve built in hull of 2D projection of 3D truncated pyramid

V. CONCLUSION

Bezier curves are very interesting tool for computer graphics, animation and game design. In this work were invented e-application to show process of rendering this type of parametric curves and told about major historical and mathematical basics and how it possible to use them in life. So, Bezier curves are not just mathematical thing for modeling smooth curves, they became important part of modern graphical design, and most of people face up with these curves everyday when they open modern web-sites, play computer games and even look the weather or cloud drive (sign of cloud developed with Bezier curves).

To upload and try to use this program, go to the link below on GitHub: https://github.com/akurganskij/E-application-for-rendering-Bezier-curves

VI. REFERENCES

1. Gerald Farin. Curves and Surfaces for CAGD A Practical Guide Fifth Edition. Arizona, MORGAN KAUFMAN N PUBLISHERS, 1999. – 499 p.
2. Michael E. Mortenson. Mathematics for computer graphics application. NY, Industrial Press, Inc., 1999. – 354 p.
3. https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcT3nGmKA-pGWh47OuMUvTKhG7hfTuQbdMG3SQ&usqp=CAU
4. https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRE-y3IzQO61qrh5Jv10-FwoPh_0yGK6A3QfQ&usqp=CAU
5. https://images.huffingtonpost.com/2012-06-08-angryBirds2.png

INVESTIGATION OF THE INFLUENCE OF EXTERNAL FACTORS ON THE POTENTIAL PERFORMANCE OF A PERSON AT THE COMPUTER AND HIS BRAIN ACTIVITY

Authors: ***Aleksandr Marchuk, Yaroslav Davydov***
Advisors: ***Liudmyla Vasylieva, Ihor Staskevych***
Donbass State Engineering Academy (Ukraine)

Abstract. *This study aims to create tools to maximize the potential performance of a person at the computer. The effects of mental fatigue on efficiency and, consequently, on productivity is difficult to overestimate. People who work a lot at the computer are particularly prone to high cognitive load and, as a result, loss of motivation and poor performance. In this work, experiments were performed to change the usual parameters of the environment during the working day and investigated their impact on the efficiency of human interaction with the computer and its electrical activity of the brain. The hypothesis of a correlation between electroencephalographic data collected using a portable neurointerface and mental fatigue has been partially confirmed. The created software-hardware system based on the obtained results will be a useful tool for optimizing the working conditions of employees in the IT field and others.*

Keywords: *EEG, programming, mental fatigue, neurointerface, statistic analysis, efficiency.*

I. INTRODUCTION

Due to the technological boom in the last two decades, the percentage of jobs for people who are exposed to cognitive stress on their brain has increased many times, outweighing the percentage of jobs with physical activity. There is a growing demand for high-quality software to solve various problems in our world. At the same time, the profession of a programmer is currently one of the most paid and one

International Competition of Student Scientific Works

BLACK SEA SCIENCE 2021

Information Technology, Automation and Robotics

Proceedings

Odessa National Academy of Food Technologies

The collection includes student works of the participants of the competition, which were not included in the number of prize-winners. The texts of the competitive works are published in the form in which they were submitted by the authors. The authors of the articles are responsible for the content and form of submission of the material.

Responsible for the issue: Sergii Kotlyk

Computer typesetting and layout: Oksana Sokolova

Odessa 2021