IB Subject: Mathematics

Extended Essay

The analysis of Usage of Bezier Curve in Animation

Research Question: How Beziers Curve used in animation

Word Count:2.769

Table Of Contents

Introduction

From the first time I have met video games (which is back in first grade) I have been obsessed with them and I have developed an interest in the area of computer science especially in the development of games. One of the fastest developing and for me one the most important aspect of games is the quality of animation. As the time passed by, the smoothness of the animation in the games have developed severely, so I wanted to learn more about animating in virtual planes and the mathematics behind this. While researching the topic I encountered the Bezier curve which had a really important place in animation and as a result I wanted to learn more on the topic.

The first examples of animation were from French inventor Charles-Émile Reynaud who developed projection praxinoscope and with it created “Théâtre Optique”. The films he made had 300 to 700 frames for a movie that was 10 to 15 minutes. Right now 60 frames per second is a standard in the industry. After that Lumiere and his Cinematograph came in. To animate Marie-Georges-Jean Méliès colorized the films by hand and also used stop trick which gave birth to stop motion animation later on in history. In 1910’s larger animation studios came into being but animation and drawings were still hand drawn. Computer animation’s earlier examples were given in 1940’s to mid 1960’s. After these earlier examples computer generated animation started spreading and dominated the field as it was a lot cheaper and faster compared to hand drawn. As with everything in computers and computing these computer-generated images and animations involve a heavy usage of mathematics and one of the most used things is the Bezier Curve.

Bezier Curve was widely used in 1960's and was invented by French engineer Pierre Bezier who used it to develop automobiles for Renault. Even though it was widely used in 1960's its basis was invented in the start of 20th century which is the Bernstein polynomials.

Bezier curves have many usages and, in many areas, related to virtual animation such as in animation, computer graphics, bounding boxes in games. It is used in programs like Photoshop and Adobe Illustrator.

Interpolation

Interpolation is the estimation of a value of a function from its known values and in Bezier curves interpolation, specifically linear interpolation is widely used.

Linear Interpolation

Linear Interpolation is done when known points are connected with straight lines and values are estimated that way. To do this slope is used since they are on the same line, they should have same slopes which can be shown as:

$$\frac{y_1 - y_0}{x_1 - x_0} = \frac{y - y_0}{x - x_0}$$

Bezier Curve

Bezier Curve is a parametric curve which is used to draw curves with all kinds of shapes and is controlled by its control points. There are types of Bezier Curve's which are named by the number of control points and when there is one it is called linear Bezier Curve, when there is two it is referred as quadratic Bezier Curve and lastly when there is three it is referred as cubic Bezier Curve.

Linear Bezier Curve

In linear Bezier Curves number of control points is one and is referred as P_1 and starting point is referred as P_0. Linear Bezier curve is also a linear interpolation between $P_0$and P_1. A linear Bezier Curve is represented in the figure below and is written with the equation: $l(t) = (1 - t)P_0 + tP_1$. The t values vary between 0 and 1. The given purple point represents the point when t is equal to 0.25 while the yellow point represents when t is equal to 0.5. Because it can also be represented with linear interpolation all of their slopes are equal as stated before. So when t is 0.25, coordinates of P_0 is $(-2.94, -1.43)$ and coordinates of P_1 is $(4.78, 2.804)$ the equation of green point is: $l(0.5) = [(1 - 0.5)(-2.94) + 0.5(4.78), (1 - 0.5)(-1.43) + 0.5(2.804)]$ which is equal to $(0.92, 0.687)$. If you calculate the slope using formula $\frac{y_1 - y_0}{x_1 - x_0}$ the slope is 0.548 this should be equal to $\frac{y - y_0}{x - x_0}$ and when the equation is solved, we would again get the result 0.548, thus we can prove that this is a linear interpolation.

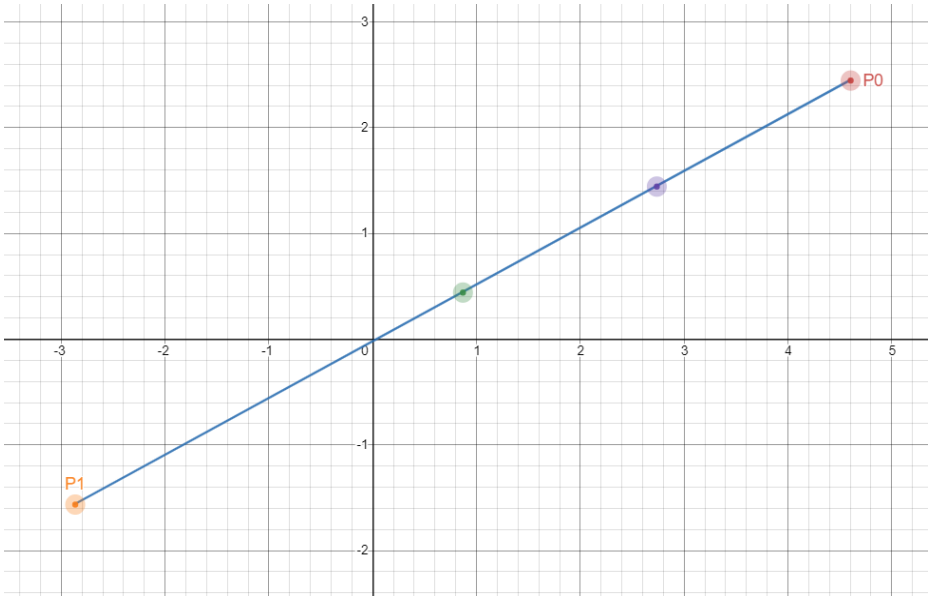

Figure 1. *Shape of a linear Bezier Curve*

Quadratic Bezier Curve

In quadratic Bezier Curves number of control points are two and are referred as P_1 and P_2. The figure below represents a quadratic Bezier Curve. The equation of the curve is:

$q(t) = (1 - t)^2 P_0 + 2(1 - t)tP_1 + t^2 P_2$ where the t values vary between 0 and 1 as same as the linear. In quadratic Bezier curves there are linear interpolations between P_0 and P_1 and also between P_1 and P_2 these interpolated points are also linearly interpolated between each other which gives us the shape of the curve in the figure 2 and 3. In figure 3 the interpolation between the interpolated lines is shown with the orange line.

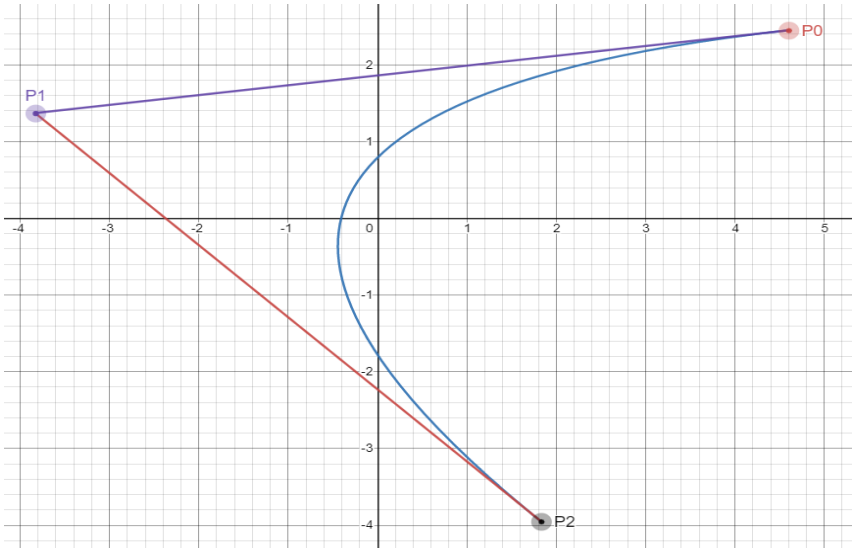

Figure 2. *Shape of a quadratic Bezier curve*

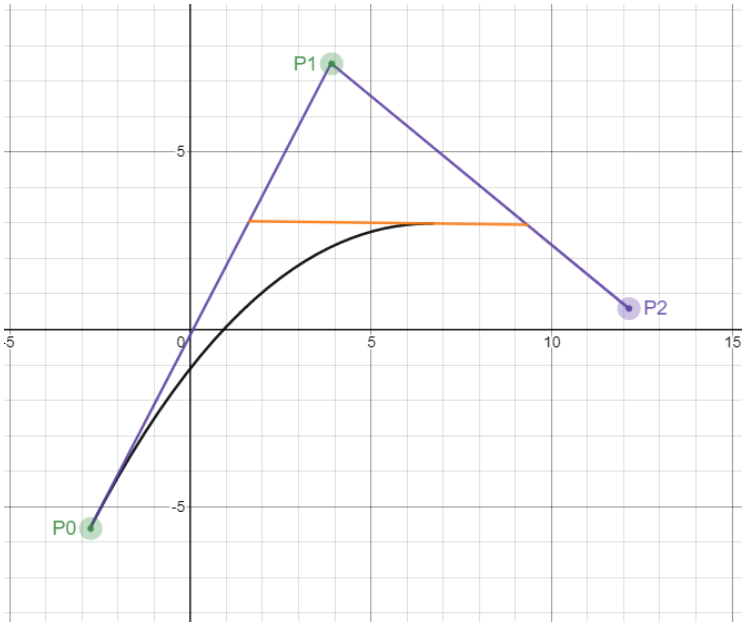

Figure 3. *Shape of a quadratic Bezier curve with interpolations given*

Cubic Bezier Curve

Cubic Bezier Curve is the most commonly used Bezier Curve and is controlled by three control points which are represented by P_0, P_1,P_2 and P_3. The equation that gives the curve is:

$c(t) = (1 - t)^3 P_0 + 3(1 - t)^2 (t) P_1 + 3(1 - t)(t)^2 P_2 + t^3 P_3$ As stated in the quadratic Bezier Curve the quadratic Bezier curve is an interpolation of interpolations and in the cubic Bezier Curve the shape is given by the interpolation of interpolation of neighboring lines such as P_0 and P_1. To simplify, straight lines are drawn to connect points $P_0$and P_1,P_1 and P_2, P_2 and P_3. After those lines P_0, P_1 and P_1, P_2 are connect with a straight line to each other same process is done with P_2, P_3 and P_1, P_2 and these straight lines are connected to each other which gives us the shape in figure 4 and in figure five shape with the interpolations is given.

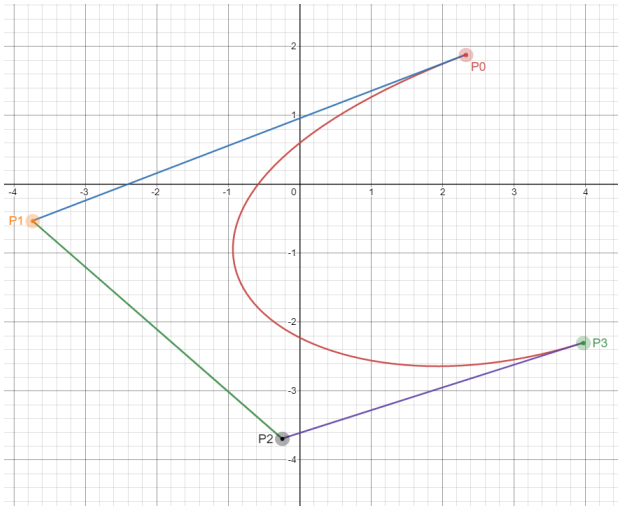

Figure 4. *Shape of a cubic Bezier curve*

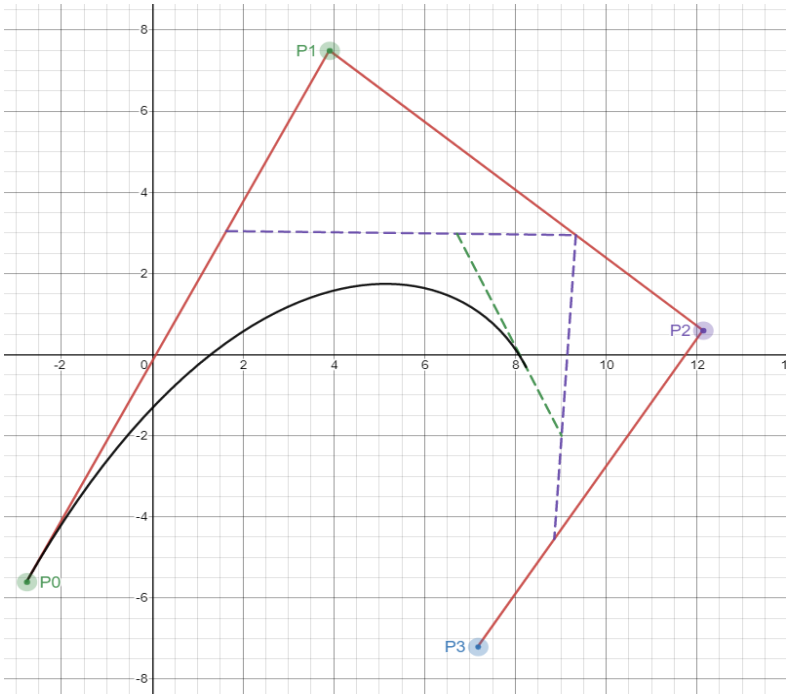

Figure 5. *Shape of a cubic Bezier curve with interpolations shown*

Derivative Of Bezier Curve

As the values of t give position of the point. When we use the derivative of a Bezier Curve it gives us the rate of change of its position which is the velocity of the point. If we use the second derivative it gives us the acceleration of the point. Which I will use more in the following part to adjust the speed of the animation.

Using Bezier Curve to Animate

To demonstrate how Bezier curves are used in animating I have used multiple tools, first one being Desmos. As I stated in the last paragraph the first derivative of the Bezier curve gives the velocity of the point, to demonstrate the effect of this, I have used Desmos and showed it on a linear Bezier curve. The figure given below shows the same linear Bezier in figure 1 but I have changed the coefficient of the t to 10. Which made the curve 10 times longer but as the t values

are stuck between 0 to 1 it makes t value reach 1 times faster thus increasing the velocity 10 times. The derivative of the linear Bezier curve is written as:

$$l(t) = (1 - t)P_0 + tP_1$$

$$l'(t) = -P_0 + P_1$$

and for the values I used derivative equals to 7.74 and when the coefficient of t is 10 the derivative is $f'(t) = -10P_0 + 10P_1$ and the value is 77.4 as stated, it is 10 times larger thus the velocity is 10 times larger.

The quadratic Bezier curve's derivative is:

$$q(t) = (1 - t)^2P_0 + 2(1 - t)tP_1 + t^2P_2$$

$$q(t) = \left(P_0 - 2tP_0 + t^2P_0\right) + \left(2tP_1 - 2t^2P_1\right) + t^2P_2$$

$$q'(t) = -2P_0 + 2tP_0 + 2P_1 - 4tP_1 + 2tP_2$$

and the derivative of the cubic Bezier Curve is:

$$c(t) = (1 - t)^3P_0 + 3(1 - t)^2(t)P_1 + 3(1 - t)(t)^2P_2 + t^3P_3$$

$$c(t) = \left(-t^3 + 3t^2 - 3t + 1\right)P_0 + \left(3t^3 - 6t^2 + 3t\right)P_1 + \left(3t^3 + 3t^2\right)P_2 + t^3P_3$$

$$c'(t) = P_0\left(-3t^2 + 6t - 3\right) + P_1\left(9t^2 - 12t + 3\right) + P_2\left(-9t^2 + 6t\right) + P_3\left(3t^2\right)$$

If we use the second derivative as stated in the last paragraph it gives us the acceleration and in linear Bezier curve acceleration is zero it goes with a fixed velocity but for both quadratic and cubic Bezier curve there is acceleration. The equation for the second derivative of the quadratic

Bezier curve is: $q''(t) = 2P_0 - 4P_1 + 2P_2$

For cubic Bezier Curve is: $c''(t) = P_0(-6t + 6) + P_1(18t - 12) + P_2(-18t + 6) + P_3(6t)$

Figure 6. *When coefficient of t is 10 rather than 1 in a linear Bezier curve*

The other tool I used is Visual Studio Code (VSC) and I used HTML (Hypertext Markup Language), CSS (Cascading Style Sheets) and also Java Script and the codes are included in the appendix. The figure below shows how the final product looks like and as the velocity increases the time to travel decreases but distance stays the same. With the slider the value of velocity could be changed and run button runs the animation to show how fast the ball moves.

Figure 7. *Bezier Curve Animation Application*

Conclusion

In this essay I have explored the effect of Bezier Curve on animation. With the existence and spreading of computers, they became part of every area and thus our lives. One of the areas that was highly affected is the animation industry. Before the computers animators would work day and night to draw all the frames of the animation to create their works. After the revolution computers created rather than animators working day and night to create works computers would do the same work in much more minimal time and with lot less money because of these reasons use of computers spread really fast and the way animation works in computers involves lots of mathematics like Bezier Curve which is the main topic of this essay. Bezier Curve is a parametric curve found by Pierre Bezier in 1960's to design cars for Renault but later it had much more uses. Curve can be used to draw anything as it is adjustable by its control points. The number of control points is how the Bezier curves are named when there is one control point present the name of it is linear Bezier Curve, when there is two is called as quadratic Bezier Curve and when there is three it is referred as cubic Bezier curve which is the mostly used as it can be adjusted easily and given lots of shapes. The main principle behind these curves is interpolation. With the shape of the curves the smoothness of the movement can be adjusted easily and when derivative is used the velocity can be found which shows how fast the animation move. In this essay I have animated a balls movement on a Bezier Curve and adjusted it speed through using its derivative.

In conclusion Bezier curves have a really big place in computer animation as they can change many aspects like the shape, smoothness or speed of the animation and with this many uses packed with the ease of use it is quite easy to encounter Bezier curve in computer animations.

Bibliography

The beauty of Bézier curves - youtube. (n.d.). Retrieved March 3, 2022, from https://www.youtube.com/watch?v=aVwxzDHniEw

Bezier curves - quadratic & cubic. Desmos. (n.d.). Retrieved March 12, 2022, from https://www.desmos.com/calculator/lvdgnyhkvy?lang=tr%2C

Encyclopædia Britannica, inc. (n.d.). *Interpolation*. Encyclopædia Britannica. Retrieved March 10, 2022, from https://www.britannica.com/science/interpolation

Khan Academy. (n.d.). *Animation with bezier curves | introduction to animation | animation | Pixar in a box | computing*. Khan Academy. Retrieved March 8, 2022, from https://www.khanacademy.org/computing/pixar/animate/ball/pi/animation-with-bezier-curves

Movement on Bezier Curve . (n.d.). Retrieved March 10, 2022, from https://codepen.io/LFCProductions/pen/YzNBgEd

Wikimedia Foundation. (2022, February 17). *Bézier curve*. Wikipedia. Retrieved February 20, 2022, from https://en.wikipedia.org/wiki/B%C3%A9zier_curve#:~:text=A%20B%C3%A9zier%20curve%20(%2F%CB%88b,by%20means%20of%20a%20formula.

Wikimedia Foundation. (2022, March 15). *History of animation*. Wikipedia. Retrieved February 25, 2022, from https://en.wikipedia.org/wiki/History_of_animation

Appendix

HTML

```
<div id="application">
    <h1>Bezier Curve Animation</h1>
    <canvas width="1800" height="1200" id="canvas"></canvas>
    <button id="play-btn">Run</button>
    <div id="Velocity-slider">
        <p>Velocity <span id="slider-text">50</span></p>
        <input type="range" min="1" max="10" value="50" id="slider" />
    </div>
</div>

<script src="script.js"></script>
```

CSS

```
* {
    box-sizing: border-box;
    font-family: Georgia, Helvetica, serif;
}

body {
    margin: 0;
    background-color: #333;
```

```
9.  }
10.
11. #application {
12.     display: flex;
13.     flex-direction: column;
14.     justify-content: center;
15.     align-items: center;
16.     color: white;
17.     text-align: center;
18.     min-height: 100vh;
19. }
20.
21. #canvas {
22.     border-radius: 5px;
23.     background-color: #f0f0f0;
24. }
25.
26. #play-btn {
27.     margin-top: 20px;
28.     outline: none;
29.     border: none;
30.     background-color: #d66e19;
```

```
    color: white;
    font-size: 30px;
    padding: 10px 30px;
    border-radius: 5px;
    cursor: pointer;
}

#play-btn:hover {
    background-color: #e60000;
}

#play-btn:focus {
    box-shadow: 0 0 0 4px rgba(179, 0, 0, 0.5);
}

#Velocity-slider {
    text-align: center;
    font-size: 30px;
    margin-top: 10px;
}

#Velocity-slider p {
```

```
    margin: 0;
}

#slider {
    margin-top: 10px;
    appearance: none;
    width: 300px;
    border-radius: 5px;
    height: 25px;
    background: white;
    outline: none;
}

#slider::-webkit-slider-thumb {
    appearance: none;
    border: none;
    width: 25px;
    height: 25px;
    border-radius: 50%;
    background: #ff0000;
    cursor: pointer;
}
```

```
75.
76. #slider::-moz-range-thumb {
77.     width: 25px;
78.     border: none;
79.     height: 25px;
80.     border-radius: 50%;
81.     background: #ff0000;
82.     cursor: pointer;
83. }
```

Javascript

```
1. const canvas = document.getElementById("canvas");
2. const ctx = canvas.getContext("2d");
3.
4. let playBtn = document.getElementById("play-btn");
5. let playAnim = false;
6.
7. let mousePos = null;
8.
9. let slider = document.getElementById("slider");
10. let sliderTxt = document.getElementById("slider-text");
11.
```

```
12. sliderTxt.textContent = slider.value;
13. slider.oninput = () => {
14.     sliderTxt.textContent = slider.value;
15.     parseSliderValue(slider.value)
16. }
17.
18. function parseSliderValue(sliderValue) {
19.     let tPercentage = sliderValue / 10;
20.
21.     tPercentage = tPercentage * 0.1;
22.     ball.speed = tPercentage;
23. }
24.
25. function playBtnText() {
26.     if(ball.x === points[3].x && ball.y === points[3].y){
27.         playBtn.textContent = "Restart?";
28.         slider.disabled = false;
29.     }
30. }
31.
32. let ball = {x:30,y:30,speed:0.1,t:0,radius:20};
33.
```

```
let points = [
    {x:ball.x,y:ball.y},
    {x:40,y:600},
    {x:1740,y:50},
    {x:1730,y:900}
]
let posRadius = 7;
let pointToMove = null;

let isClickDown = false;

function moveBallInBezierCurve() {
    let [p0, p1, p2, p3] = points;

    let cx = 3 * (p1.x - p0.x);
    let bx = 3 * (p2.x - p1.x) - cx;
    let ax = p3.x - p0.x - cx - bx;

    let cy = 3 * (p1.y - p0.y);
    let by = 3 * (p2.y - p1.y) - cy;
    let ay = p3.y - p0.y - cy -by;

```

```
    let t = ball.t;

    ball.t += ball.speed;
    let xt = ax*(t*t*t) + bx*(t*t) + cx*t + p0.x;
    let yt = ay*(t*t*t) + by*(t*t) + cy*t + p0.y;

    if(ball.t > 1){
        ball.t=1;
    }

    ball.x = xt;
    ball.y = yt;
    drawBall();
}

function drawBall() {
    ctx.fillStyle = "black";
    ctx.beginPath();
    ctx.arc(ball.x,ball.y,ball.radius,0,Math.PI * 2,false);
    ctx.fill();
}
```

```
78. function drawPoints() {
79.     ctx.fillStyle = "red";
80.     points.forEach(point => {
81.         ctx.beginPath();
82.         ctx.arc(point.x,point.y,posRadius,0,Math.PI * 2,false);
83.         ctx.fill();
84.
85.         ctx.font = "11px Arial";
86.         ctx.fillText(`(${point.x},${point.y})`,point.x,point.y+30);
87.     });
88. }
89.
90. function isMouseOverPoint(point) {
91.     let dx = mousePos.x-point.x;
92.     let dy = mousePos.y-point.y;
93.     return(dx*dx+dy*dy<posRadius*posRadius);
94. }
95.
96. function checkIfCursorInPoint(){
97.     if(mousePos && isClickDown && !pointToMove){
98.         points.forEach(point => {
99.             if(isMouseOverPoint(point)){
```

```
                pointToMove = point;
            }
        })
    }
}

function movePoint() {
    if(pointToMove === points[0]){
        points[0].x = mousePos.x;
        points[0].y = mousePos.y;
        ball.x = mousePos.x;
        ball.y = mousePos.y;
        return
    }
    let pointIndex = points.indexOf(pointToMove);
    points[pointIndex].x = mousePos.x;
    points[pointIndex].y = mousePos.y;
}

function drawLine() {
    ctx.beginPath();
    ctx.setLineDash([8, 15]);
```

```
122.        ctx.moveTo(points[0].x,points[0].y);
123.        ctx.bezierCurveTo(points[1].x, points[1].y, points[2].x, points[2].y, points[3].x,
    points[3].y);
124.        ctx.stroke();
125.    }
126.
127.    function animate() {
128.        requestAnimationFrame(animate);
129.        ctx.clearRect(0,0,canvas.width,canvas.height);
130.        playBtnText();
131.
132.        if(!playAnim){
133.            drawBall();
134.        }else{
135.            moveBallInBezierCurve();
136.        }
137.        if(!slider.disabled) checkIfCursorInPoint();
138.        if(pointToMove) movePoint();
139.        if(!slider.disabled) drawLine();
140.
141.        if(!slider.disabled) drawPoints();
142.    }
```

```
143.
144.    animate();
145.
146.    playBtn.addEventListener("click", () => {
147.        playAnim = true;
148.        slider.disabled = true;
149.        if(ball.x === points[3].x && ball.y === points[3].y){
150.
151.            ball.t = 0;
152.            ball.x = points[0].x;
153.            ball.y = points[0].y;
154.
155.            playBtn.textContent = "Play";
156.        }
157.    });
158.
159.    canvas.addEventListener("mousemove", e => {
160.
161.        mousePos = {
162.            x: e.clientX - canvas.offsetLeft,
163.            y: (e.clientY - canvas.offsetTop) + scrollY
164.        }
```

```
165.    });
166.
167.    canvas.addEventListener("mousedown", () => {
168.        isClickDown = true;
169.    });
170.
171.    canvas.addEventListener("mouseup", () => {
172.        isClickDown = false;
173.        pointToMove = null;
174.    });
175.
176.    parseSliderValue(slider.value);
```