STAG: Smart Tools and Applications in Graphics (2020)
S. Biasotti, R. Pintus and S. Berretti (Editors)

Curve Design Studio: Bézier curve integrated tool for video game development

L. C. Viagrande [1] and D. Allegra [1] and F. Stanco [1]

[1]University of Catania, Catania, Italy

Abstract

Bézier curve is a useful tool for computer graphic applications. A curved road, a car, a roller coaster are just few examples of objects that could be described by a Bézier curve. The video game industry already used this kind of curves in their products around the late 90', but modern game engines suffer from a lack of integrated plugin capable of manipulating and exploiting the full potential of a curve. Usually, modern game engines rely on external tools for addressing curves manipulation or they just solve some specific problems related to curves, such as animation control. In this paper we propose Curve Design Studio, a novel tool embedded in `Unity3D Game Engine` *for assisting users to solve the main problems related to the use of curves, that may arise during the making process of a video game or any other application made with that game engine. This tool, gives the opportunity to create Bézier curves easily and adapt them to many tasks, by avoiding the use of multiple external tools which solve just one specific problem. Consequently, it guarantees time-saving and lower-complexity in projects development.*

CCS Concepts

• ***Software and its engineering*** → ***Software creation and management;*** • ***Mathematics of computing*** → *Solvers;* • ***Computing methodologies*** → ***Graphics systems and interfaces;***

1. Introduction

Computer graphics tries to represent reality in a virtual and simulated environment. This led the researchers to propose new algorithms for realistic image synthesis. However, the realism we have at the present day could not have been achieved without the exploitation of specific mathematical instruments [Car16, Dob19].

A mathematical concept widely used in computer graphics, especially in video games industry, is parametric curve [Han11, Raj20]. The use of parametric curves is needed in many aspects of the video game development process like path definition, car routes, 3D modelling and so on. In the early 1960s, Peter Bézier began researching a better way to represent curves and surfaces that could have been useful for automotive engineers. He was familiar with cubic curves and bicubic surfaces researched by Ferguson and Coons [Sal06]. However, these methods had drawbacks, especially when shapes alteration and control was required. Bézier Curves were proposed to face this problem and became part of the UNISURF system, used by the French automobile manufacturer Renault to design the surfaces of many of their products.

Due to its simple geometric construction, Bézier curves are widely used in many fields, and, with the growth of computer graphics applications, it was an opportunity to introduce it as a tool for vector drawing software [Far02]. This allows to realise high precision drawings with little effort. Nowadays, computer graphics programs and design systems provides tools for Bézier Curve manipulation, so that they are used for many practical applications. They are even employed for designing the shape of letters in fonts and for many other graphical and practical applications [Mor99, Sch19]. In the past decades, the video game industry focused on the use of Bézier curves and their derivatives for the representation of 3D graphical assets in order to hide defects that emerge from a polygonal realisation [Han11]. They have also been used in animations and time driven events [Erl02, Kha11]. We have evidence of curves employment in game engines like `Unity3D`, but they are usually coded for specific purpose [Ber15, WU,16]; alternatively, curves are used in third-part software [Haw13, Cha15, DIA18] for creating 3D objects, like roads and other curve shaped object which are subsequently imported in game engines. It is possible to generate curves in other engines such as `Unreal engine` but only with coding, without the possibility of customization already present in other tools. We have evidence of a conceptual use of Bézier curves for terrain development [De 11] and also an application developed on `Unity` for the construction of terrains by using curves [de 16]. Other software improve the `Unity particle system` by exploiting the Bézier curves [Des16]. Nevertheless, the programmer is usually forced to program their curve manually. This lack of tools and algorithms for using curves and exploiting their full potential has motivated our work.

DOI: 10.2312/stag.20201236

In this paper we propose "Curve Design Studio", a novel tool embedded in Unity3D Game Engine for visual representation of a Bézier curve and for its application in many tasks of video game development process. The use of a plugin embedded in a game engine capable of manipulating a curve may save video games companies financial resources and time. Indeed, they may not be constrained to use other computer graphic applications for those precision tasks. The proposed tool implements smart algorithms to recreate a Bézier curve in the Unity3D virtual environment without the need to rely on external software. The proposed tool allows to assign custom parameters to the curve and the related points and real time changing. To prove the validity of the proposed work, at the end of the paper we present a custom solution for the movement of an object along the curve. The rest of the paper is organised as follows: in Section 2 we state the problem and analyse previous works. In Section 3 we introduce Curve Design Studio. In Section 4 we analyse quality and efficiency of the proposed plugin through the use of a custom script: FollowPath. It uses the proposed tool for moving a 3D object in the virtual environment. In Section 5 we report the results and our conclusions.

2. Statement of the problem and previous works

Our work is motivated by the lack of Unity3D tools for using Bézier curve in many aspects of a video game development process. This is a well-known problem to the community of game developers. Over the years, the community has created many auxiliary tools capable of manipulating a Bézier curve in order to meet these lacks. Unfortunately, each tool addresses a specific problem. Video game developers are hence forced to use multiple software thus increasing project complexity and development time. Some tools to create Bézier curves are hence present within the Unity3D asset store. Among them: *Bézier Curve Editor, Bézier Master, Spline Mesh, BG Curve* [Ark, Ban, Ale, Ben]. However they all focus on a limited videogame development aspects and do not offer a comprehensive overview of what could be done by using curves; moreover, they have a too slow learning curve and are not recommended for those without programming skills. Bézier Curve Editor permits an easy creation and modification of Bézier splines. Such spline can be used as a path for game objects and as animation curve. The toolkit also has APIs for getting various spline info, but they are minimal. Bézier Curve Editor has a big limitation: it is not possible to edit or generate a spline in Game. SplineMesh is a powerful tool with an intuitive editor and excellent performance, great for creating the following content:

- A Bézier spline. (However, it is not possible to retrieve much mathematical information);
- 3D Mesh extrusion, bending or cloning through a spline;
- 2D Mesh extrusion along the spline.

SplineMesh does not have the flexibility of Curve Design Studio from a mathematical point of view but it remains, however, an excellent tool. BG Curve is one of the most complete curve-related asset package. First of all, it is possible to generate and modify a Bézier spline both in real time and in the scene view and a lot of mathematical information can be obtained from the generated spline, thanks to several dedicated APIs. BG Curve has many additional features: the creation of a mesh, the movement of game objects, the creation of 3D or 2D colliders, the cursors and so on. However, given the complexity of the mentioned plugin, Curve Design Studio turns out to be more practical and easier to use.

The proposed tool addresses several problems related to the use of a curve in a video game development process such as:

- animating an object along a path;
- create paths and terrain along a curve;
- instantiating objects on the points of a curve;
- extrude an object along a curve;
- creating complex behaviours for artificial intelligence agents (patrolling, car driving, etc).

Moreover, it aims to provide a framework which can be easily extended.

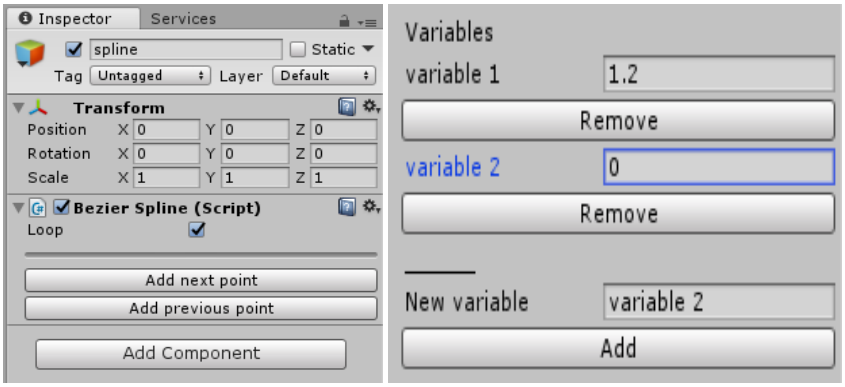
3. Method

Unity3D engine is an environment which integrates a game engine provided by Unity Technologies. Unity is typically employed to produce digital games for different platform, such as PC's, consoles, mobile devices and websites. It allows to handle 3D model and other kinds of assets, such as materials, lights, images, and videos. Unity gives the possibility to encode algorithms in two different programming languages: C# and JavaScript. Although Unity is often used for digital game development, it could be employed for generic purpose application related to 3D scenes a models. The main feature of Unity is the convenient way to manage multimedia data, the user-friendly development GUI and the multi-platform builder [Xie12].

In this work we propose a novel tool called Curve Design Studio as plugin for Unity3D. Our tool implements the concept of Bézier curve in a three-dimensional space, giving developers the chance to use the curve without binding it to a specific problem. It can be used to address multiple issues which usually occur during the development of a visual application. The algorithm allows the construction of a Bézier curve through its anchor and control points. An example of a standard linear Bézier curve in the Unity virtual environment, where all points are aligned, is shown in Fig. 2. This curve is automatically created in the Unity editor when the script Bézier Spline is attached inside an empty gameobject previously inserted in the Unity editor scene, as it is shown in Fig. 1(a). The user can manipulate the curve through the control points, represented by the smaller squares. Indeed, selecting these points the Unity visual manipulation system will appear, represented by the presence of the three Cartesian axes. The bigger squares represents the anchor points.

Hence, it is possible to manipulate these points in the three-dimensional space and to assign some custom variables to them. Moreover, the algorithm extends the visual editor of Unity and provides useful tools for curve manipulation even in game-mode. What is permitted in this case is:

- to close the spline with a check box called loop;
- to add other points to the curve;
- to move a selected point;
- to delete a selected point;
- to restore a deleted point.



(a) (b)

Figure 1: *(a) GameObject with Bézier Spline script attached showing customization elements within* `Unity editor`*; (b) Custom variables for anchor points;*

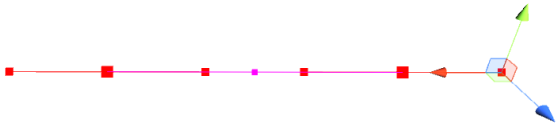


Figure 2: *Standard linear Bézier curve where anchor and control points lie along the same line.*

A more complex curve, realised with the help of the extended visual editor, is shown in Fig. 3.

Additionally, by moving the points, it is possible to change their own behaviour with an attribute called *Style*. Three styles for each point can be selected:

- Free style. The control points are unbounded. (see Fig. 4(a))
- Mirror style. Control points will form a line with the anchor point and must be at the same distance from it. (see Fig. 4(b))
- Align style. Control points will form a line with the anchor point but they can have different distance from it. (see Fig. 4(c))

In addition, as mentioned before, it is possible to assign parameters to any point in the spline. When selecting one of the available points, the `Unity editor` will show a control window where the user can add variables and remove them as shown in Fig. 1(b) To remarks the advantages of the proposed tool, we show an example application creating by using Curve Design Studio: Follow-path. It shows how a 3D object could be easily moved, in the three-dimensional environment, along the Bézier curve.

Moreover, it shows how to manipulate its parameters, i.e. the scale factor and the custom variables assigned to the curve points.

3.1. Preliminary

Here, we provide mathematical notions of a Bézier curve, and how to reach its graphical representation through the De Casteljau's algorithm.

3.1.1. Bézier curve representation

Given three points, A, B, C, so that line in $\overline{AB}$ is tangent to the curve at A and line $\overline{BC}$ is tangent to C, the curve begins at A and ends at C. For any ratio u_i, where $0 \leq u_i \leq 1$, the points D and E are constructed so that

$$\frac{\overline{AD}}{\overline{AB}} = \frac{\overline{BE}}{\overline{BC}} = u_i \tag{1}$$

On $\overline{DE}$, F is constructed so that

$$\frac{\overline{DF}}{\overline{DE}} = u_i \tag{2}$$

Hence, the point F belongs to the curve. For other values of u_i, a series of points are produced on a Bézier curve (see Fig. 5) [Mor99]. *These points are called control points. The first and last point on the curve are called anchor points.*

3.1.2. De Casteljau's algorithm

A Bézier curve Be of grade n, with structural points $P_0 \ldots P_n$ could be written in Bernstein form as follows:

$$Be(T) = \sum_{i=0}^{n} P_i b_{i,n}(t) \tag{3}$$

where $0 \leq t \leq 1$ and b is Bernstein polynomial [Lor13]

$$b_{i,n}(t) = \binom{n}{i} (1-t)^{n-i} t^i \tag{4}$$

Nevertheless, it is not recommend directly using the equation (3)

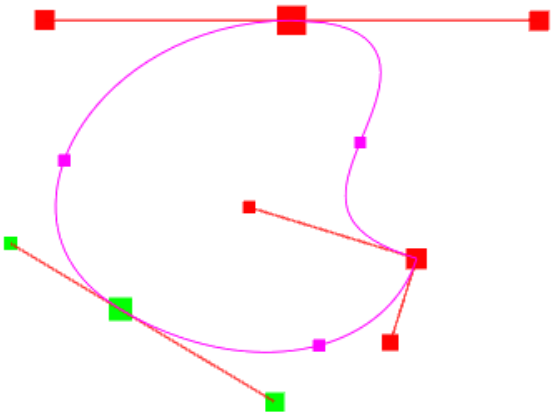
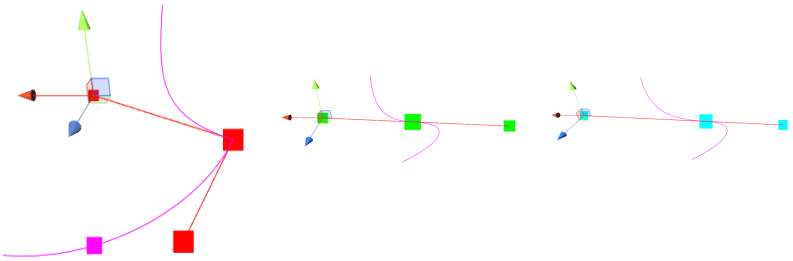


Figure 3: *Composite Bézier curve*



(a) (b) (c)

Figure 4: *(a) Free style; (b) Mirror style; (c) Align style;*

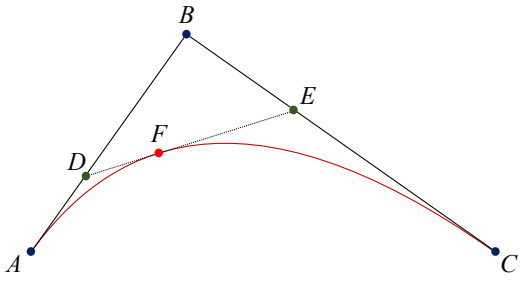


Figure 5: *Second-degree Bézier curve*

to generate a curve in a digital system, as it is not numerically stable. This pushes researchers to find a more stable strategy to draw a Bézier curve in such systems. De Casteljau's algorithm is an effective, recursive and stable method to evaluate Bernstein polynomials or Bézier curve [Pra13]. It is robust methods which gives insight into Bézier curve behaviour and leads to important operations on the curves.

According to De Casteljau's algorithm, one can evaluate a Bézier curve at t_0 with the following recurrence formula

$$P_i^{(0)} := P_i \quad i = 0, \ldots, n \tag{5}$$

$$P_i^{(j)} := P_i^{(j-1)}(1-t_0) + P_{i+1}^{(j-1)} t_0 \quad i = 0, \ldots, n \quad j = 1, \ldots, n \tag{6}$$

$$Be(t_0) = P_0^{(n)} \tag{7}$$

Hence, the recursive definition led the following equation, which is equal to the Bernstein form:

$$Be(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i P_i \tag{8}$$

where $P_0, P_1 \ldots P_n$ are control points [Roc96, Kha18].

Using De Casteljau's algorithm we represent the Bézier curves in the proposed plugin. In this work, we implement cubic Bézier curves, which are uniquely defined by 4 control points.

3.2. Curve Design Studio implementation

Here we provide a technical description of the the proposed tool.

The Cubic Bézier curve is defined into an abstract class called

Algorithm 1 Bézier

```
1: procedure GETPOINT
2: Input: P0,P1,P2,P3, t
3: Output: p4
4:     t ← clamp 0 ≤ t ≤ 1
5:     i ← 1-t
6:     p4 ← i^3*P0+3*i^2*t*P1+3*i*t^2*P2+t^3*P3
```

Bézier. The procedure is reported in Algorithm 1. Given four points $P_0 ... P_3$ and a percentage t, the method outputs the point in the curve corresponding to that percentage. To store the spatial information for each point in the curve, three lists of vectors were implemented. Two of them are used for the control points, since we have two control points for each anchor point and one list is used for the anchor points. Each point has an index.

Algorithm 2 GetNextPointIndex

```
1: Input: i
2: Output: j
3: if loop then
4:     i ← (i+1)%n_points
5:     return i
6: if i == n_points − 1 then return −1
   return ++i
```

Given an index for a point in the curve, the procedure for taking the next index is reported in Algorithm 2. *n_points* is the number of points in the list while *loop* identifies a closed curve. If a loop is found, the index gets back to 0 when it gets to the last index in the list. If not and it found the last index it returns -1. Otherwise it just returns the next index.

3.3. Point style

A list of *int* named *Style* is used to store the value of the style that can be assigned to each point. In this case, the parameter can get the following values:

- $0 = Free$
- $1 = Mirror$
- $2 = Align$

Algorithm 3 is used when a control point is moved or you have to change its style. In the case, given an index i, and a control point of index i and a $flag$, which identifies the first or the second control point, the algorithm moves the first or the second control point, specified by the parameter $flag$, taking the position of the other control point as a reference to ensure the correctness of the specified style. The method MoveToward moves the specified control point *pt* from its original position toward the position specified by the second input parameter *controlpoint* of the distance specified by the last input parameter *Distance*.

3.4. Custom Variables

As it was mentioned before, it is possible to assign one or more variables to each point in the curve. The same variable, therefore, can assume different values depending on the anchor point in which it is located. this peculiarity is very important as, in the proposed plugin, there is an algorithm that allows to obtain the interpolated value of this variable in any point of the curve between two anchor points.

Algorithm 3 FixPtStyle

```
1: Input:i,flag
2: style ← Style[i]
3: if style == 0 then return
4: if flag == 0 then
5:     if style == 1 then
6:         P2,i ← MoveToward(pti,P1,i,
7:         -Distance(pti,P1,i))
8:     if style == 2 then
9:         P2,i ← MoveToward(pti,P1,i,
10:        -Distance(pti,P2,i))
11: if flag == 1 then
12:    if style == 1 then
13:        P1,i ← MoveToward(pti,P2,i,
14:        -Distance(pti,P2,i))
15:    if style == 2 then
16:        P1,i ← MoveToward(pti,P2,i,
17:        -Distance(pti,P1,i))
```

4. Case of study

To test the quality of the analysed tool a custom script for `Unity` called FollowPath was realised. It allows to move a three dimensional object along a curve made with Curve Design Studio. Moreover it permits to modify the scale of that object exploiting some custom parameters. Thanks to Curve Design Studio this script implementation is an easy task. The only thing that is needed is a primitive three dimensional object. `Unity` provides lot of built-in objects making it easy to add a Cube in the virtual environment. After adding the cube, the script FollowPath must be linked to the object itself. `Unity` shows us some properties that we have to customize, as it is shown in Fig 6. The first thing we have to do is to link the Bézier curve to the curve variable. The object, while in play mode, can move along the linked curve, as can be seen in Fig 7. The speed variable controls the speed at which the object can move. The check box forward force the cube to orientate its z axis toward the curve direction. Last parameter is the follow path style and controls the type of movement the cube will have. A value of 2 means that the cube will move in a loop. Adding a scale variable on the points of the curve will force the cube to change its scale factor according to the value of that scale parameter.

5. Conclusion

We have presented Curve Design Studio, a plugin embedded in `Unity3D` game engine which permits the realisation of Bézier curves easily. The plugin comes with a complete suit of tools to manipulate and customise the curve and its points. Unlike the tools already present in the market, Curve Design Studio does not ad-

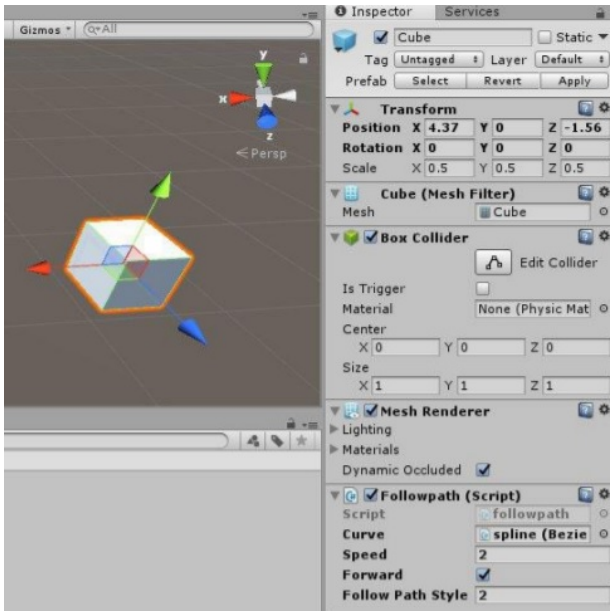


Figure 6: *Followpath configuration. To use it properly a spline must be attached in the appropriate field of the interface, assign a speed, direction and type of movement*

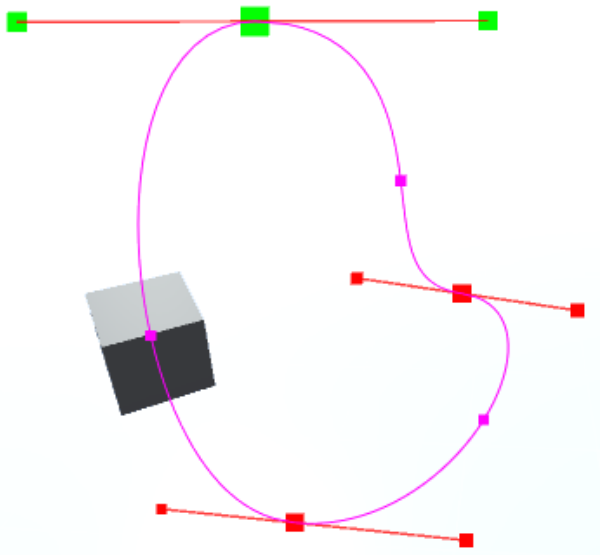


Figure 7: *Cube moving along the curve. After followpath script configuration, the cube will move along the curve drawn in purple. If variables have been added to the control points, represented by the purple squares, the cube will assume different behaviors (for example it will be scaled if there is a scale variable) once the points are reached. The other squares are anchor points; they can be used to manipulate the shape of the curve. Their colors are the standard* `Unity` *colors and indicate the direction of the selected axis*

dress any specific issue but provides the developers with a powerful tool which can be personalised and adapted to the needs that may arise during a video game development process. The versatility and strength of Curve Design Studio have been shown with FollowPath, a custom script that uses Curve Design Studio and its features to solve some issues like the movement of a three dimensional object or its scale factor manipulation. Its ease of use and its versatility make it a good tool to embed in `Unity3D` game engine replacing actual tools for curve manipulation. As future works we are organizing tests with selected groups of users with various expertise, ranging from simple users to experienced developers. In this way we will have a more accurate overview of the effectiveness and simplicity of the proposed tool.

References

[Ale] ALEXANDER KOTOV: Bézier Master. https://assetstore.unity.com/packages/tools/level-design/bezier-master-107374. 2

[Ark] ARKHAM INTERACTIVE: Bézier Curve Editor. https://assetstore.unity.com/packages/tools/bezier-curve-editor-11278. 2

[Ban] BANSHEEGZ: Bg Curve. https://assetstore.unity.com/packages/tools/utilities/bg-curve-59043. 2

[Ben] BENOIT DUMAS: Spline Mesh. https://assetstore.unity.com/packages/tools/modeling/splinemesh-104989. 2

[Ber15] BERGER, MATTHIAS AND CRISTIE, VERINA: CFD Post-processing in Unity3D. *Procedia Computer Science 51* (2015), 2913–2922. 1

[Car16] CARTER, NATHAN.: *Introduction to the Mathematics of Computer Graphics*, vol. 51. American Mathematical Soc, 2016. 1

[Cha15] CHADIMOVÁ, LENKA: The creating serious games for historical subjects at the 1st level of primary school and such as a part of exhibition chosen historical buildings. *Procedia - Social and Behavioral Sciences* (2015). 1

[De 11] DE CARLI, DANIEL MICHELON AND BEVILACQUA, FERNANDO AND POZZER, CESAR TADEU AND D'ORNELLAS, MARCOS CORDEIRO: A survey of procedural content generation techniques suitable to game development. In *2011 Brazilian Symposium on Games and Digital Entertainment* (2011), IEEE, pp. 26–35. 1

[de 16] DE TONI, RODRIGO AND MARSON, FERNANDO AND CASSOL, VINICIUS J: Dynamic terrains applying pseudo-infinity and synthesized by Bézier curves. In *Simp/'osio Brasileiro de Jogos e Entretenimento Digital* (2016). 1

[Des16] DESMOND, MICHAEL AND BAHANA, RAYMOND: Bézier VFX plug-in: Improving unity visual effect performance. In *2016 1st International Conference on Game, Game Art, and Gamification (ICGGAG)* (2016), IEEE, pp. 1–6. 1

[DIA18] DIAZ, LIZZIE EDMEA NARVAEZ AND PECH, VICTOR MANUEL CHI AND CASTRO, ERIKA ROSSANA LLANES AND EUAN, MAXIMILIANO CANCHE: Address classifier: a game-based educational application for computer networks. *International Journal of Advanced Research in Computer Science and Software ...* (2018). 1

[Dob19] DOBASHI, YOSHINORI: Mathematics in Computer Graphics. *Mathematical Insights into Advanced Computer Graphics Techniques* (2019), 1. 1

[Erl02] ERLEBEN, KENNY AND HENRIKSEN, KNUD: Scripted bodies and spline driven animation. *Graphics programming methods* (2002), 37–50. 1

[Far02] FARIN, GERALD E AND FARIN, GERALD: *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann, 2002. 1

[Han11] HAN, JUNGHYUN: *3D graphics for game programming*. Chapman and Hall/CRC, 2011. 1

[Haw13] HAWLEY, RICHARD A: *Grome Terrain Modeling with Ogre3D, UDK, and Unity3D*. Packt Publishing Ltd, 2013. 1

[Kha11] KHAN, MURTAZA ALI AND SARFRAZ, MUHAMMAD: Motion tweening for skeletal animation by cardinal spline. In *International Conference on Informatics Engineering and Information Science* (2011), Springer, pp. 179–188. 1

[Kha18] KHAN, KHALID AND LOBIYAL, DK AND KILICMAN, ADEM: A de Casteljau Algorithm for Bernstein type Polynomials based on (p, q)-integers. *Applications & Applied Mathematics 13*, 2 (2018). 4

[Lor13] LORENTZ, GEORGE G: *Bernstein polynomials*. American Mathematical Soc., 2013. 3

[Mor99] MORTENSON, MICHAEL E: *Mathematics for computer graphics applications*. Industrial Press Inc., 1999. 1, 3

[Pra13] PRAUTZSCH, HARTMUT AND BOEHM, WOLFGANG AND PALUSZNY, MARCO: *Bézier and B-spline techniques*. Springer Science & Business Media, 2013. 4

[Raj20] RAJA, SP: Bézier and B-Spline Curves - a Study and its application in Wavelet Decomposition. *International Journal of Wavelets, Multiresolution and Information Processing* (2020). 1

[Roc96] ROCKWOOD, ALYN AND CHAMBERS, PETER: Interactive curves and surfaces. In *Technology-Based Re-Engineering Engineering Education Proceedings of Frontiers in Education FIE'96 26th Annual Conference* (1996), vol. 1, IEEE, pp. 471–474. 4

[Sal06] SALOMON, DAVID: *Curves and surfaces for computer graphics*. Packt Publishing Ltd, 2006. doi:{10.1007/0-387-28452-4}. 1

[Sch19] SCHEIDERER, CHRISTIAN AND THUN, TIMO AND MEISEN, TOBIAS: Bézier Curve Based Continuous and Smooth Motion Planning for Self-Learning Industrial Robots. *Procedia Manufacturing 38* (2019), 423–430. 1

[WU,16] WU, XIAO-LIANG AND HUANG, XIANG-NIAN: Using Bézier Curve to Bend 3D Objects in Unity. *Modern Computer*, 7 (2016), 15. 1

[Xie12] XIE, JINGMING: Research on key technologies base Unity3D game engine. In *2012 7th international conference on computer science & education (ICCSE)* (2012), IEEE, pp. 695–699. 2