

졸업논문청구논문

**베지어 곡선, 베지어 곡면을 이용한 3D 모델링
개선 연구**

**Improvement of 3D Modeling Using Bezier Surface and
Bezier Curve**

박건호 (朴建皓 Park, geonho)

20046

과학영재학교 경기과학고등학교

2022

베지어 곡선, 베지어 곡면을 이용한 3D 모델링 개선 연구

Improvement of 3D Modeling Using Bezier Surface and Bezier Curve

[논문제출 전 체크리스트]

1. 이 논문은 내가 직접 연구하고 작성한 것이다.
2. 인용한 모든 자료(책, 논문, 인터넷자료 등)의 인용표시를 바르게 하였다.
3. 인용한 자료의 표현이나 내용을 왜곡하지 않았다.
4. 정확한 출처제시 없이 다른 사람의 글이나 아이디어를 가져오지 않았다.
5. 논문 작성 중 도표나 데이터를 조작(위조 혹은 변조)하지 않았다.
6. 다른 친구와 같은 내용의 논문을 제출하지 않았다.

Improvement of 3D Modeling Using Bezier Surface and Bezier Curve

Advisor : Teacher Kim, Soyeon

by

20046 Park, geonho

Gyeonggi Science High School for the gifted

A thesis submitted to the Gyeonggi Science High School in partial fulfillment of the requirements for the graduation. The study was conducted in accordance with Code of Research Ethics.*

2022. 07. 15

Approved by
Teacher Kim, Soyeon
[Thesis Advisor]

*Declaration of Ethical Conduct in Research: I, as a graduate student of GSHS, hereby declare that I have not committed any acts that may damage the credibility of my research. These include, but are not limited to: falsification, thesis written by someone else, distortion of research findings or plagiarism. I affirm that my thesis contains honest conclusions based on my own careful research under the guidance of my thesis advisor.

베지어 곡선, 베지어 곡면을 이용한 3D 모델링

개선 연구

박 건 호

위 논문은 과학영재학교 경기과학고등학교 졸업논문으로
졸업논문심사위원회에서 심사 통과하였음.

2022년 07월 15일

심사위원장 박 병 철 (인)

심사위원 서 은 아 (인)

심사위원 김 소 연 (인)

Improvement of 3D Modeling Using Bezier Surface and Bezier Curve

Abstract

Bezier surface and Bezier curve are widely used in the field of designing automobiles and aircraft bodies. The reason is that it has a very smooth shape and has little memory required for storage. This study deals with new models and RGB storage formats using Bezier surface and Bezier curve for 2D and 3D models. It was created by dividing a model and storing approximate curves and curves, referring to linear studies such as the Möller-Trumbore intersection algorithm, the least squares method, and the Gaussian Newton method. Using this method, the 3D model showed more than 100 times more efficiency than the conventional triangular splitting method, which was often used, and it could be seen with eyes that it became softer than the conventional storage method. This study showed that any model is always approximated enough when using this method. Curve and curved modeling in this study based on mathemati-

cal theoretical background and programming efficiency are expected to greatly contribute to VR, AR, and hologram fields. In addition, it will be possible to use it more in the CAGD field.

베지어 곡선, 베지어 곡면을 이용한 3D 모델링 개선 연구

초 록

Bezier surface, Bezier curve는 자동차의 차체나 비행기의 차체 등을 설계하는 분야에서 많이 사용된다. 그 이유는 Bezier 곡선, 곡면은 매우 매끄러운 형태를 지니며 저장할 때 적은 메모리가 필요하기 때문이다. 이 연구에서는 2D, 3D 모형들에 대해 Bezier surface, Bezier curve를 이용한 새로운 모델 및 RGB 저장형식을 다룬다. 이는 어떤 모형을 분할하여 근사한 곡선, 곡면을 저장하는 형식으로 Möller–Trumbore intersection algorithm, 최소제곱법, 가우스 뉴턴 방법과 같은 선형연구를 참고하여 만들었다. 이 방법을 사용해보면 3D모형은 기존에 많이 사용했던 삼각형 분할 방법보다 100배 이상의 효율을 보였고, 기존의 저장 방법보다 부드러워진 것을 눈으로 확인해볼 수 있었다. 이 방법을 이용할 때 항상 어떤 모형이든 충분히 근사 가능하다는 것을 증명하였다. 이러한 수학적 근거와 프로그래밍 효율성에 기반한 본 연구의 곡선, 곡면 모델링은 VR, AR 및 홀로그램 분야에 크게 기여할 것으로 기대된다. 또한 CAGD 분야에서 더 큰 활용이 가능할 것이다.

Contents

Abstract	i
초록	iii
Contents	iv
List of Figures	vi
I Introduction	1
I.1 Research motivation	1
I.2 Theorectical background	2
I.2.1 베지어 곡선과 곡면, 가중치 베지어 곡선과 곡면	2
I.2.2 근사 기법	4
I.2.3 경계 (Boundary)	5
I.2.4 볼록집합 (Convex set)	6
I.2.5 오차함수	6
I.2.6 obj,mtl파일	7
I.3 연구질문	9
II Main Contents	9
II.1 2D Model Improvement	10
II.1.1 분할 방법	10
II.1.2 Control Point Determination	11
II.1.3 Algorithm	12
II.1.4 수렴성 증명	14
II.2 3D model improvement	20
II.2.1 분할 방법	21

II.2.2	Control Point Determination	22
II.2.3	Algorithm	25
II.2.4	수렴성 증명	29
II.3	2D model RGB implementation	32
II.3.1	분할 방법	33
II.3.2	Control Point Determination	34
II.3.3	Algorithm	35
II.3.4	수렴성 증명	38
II.4	3D model RGB implementation	41
II.4.1	Control Point Determination	41
II.4.2	Algorithm	43
II.4.3	수렴성 증명	46
II.5	Model Movement Implementation	48
III	Code Implementation	49
III.1	Lambertian Reflectance	49
III.2	Arc Approximation Program	51
III.3	3D model approximation program	54
IV	Conclusion	72
References	74
감사의 글	75
연구활동	77

List of Figures

Figure 1.	2차 베지어 곡선	2
Figure 2.	베지어 곡선(a)과 베지어 곡면(b)	3
Figure 3.	가중치 베지어 곡선(a)과 가중치 베지어 곡면(b)	4
Figure 4.	조각별과 조각	10
Figure 5.	2차 베지어 곡선의 성질	12
Figure 6.	곡선을 조각별로 나누기	13
Figure 7.	중간점	14
Figure 8.	각 조각을 근사하기	15
Figure 9.	근사수치	15
Figure 10.	베지어 곡선의 매개화	16
Figure 11.	L 에 평행한 성분	19
Figure 12.	$(2,2)$ -차 베지어 곡면	23
Figure 13.	근사 방법	24
Figure 14.	원통형 분할	25
Figure 15.	조절점	26
Figure 16.	b_{11} 을 구하는 방법을 적용한 예시	28
Figure 17.	조절점 - b_{11}	29
Figure 18.	구면 근사	30
Figure 19.	2D RGB 분할	34
Figure 20.	원통형 분할	43
Figure 21.	베지어 곡선의 움직임	49
Figure 22.	Lambertian reflectance	50

I. Introduction

I.1 Research motivation

베지어 곡선과 베지어 곡면을 이용한 연구는 여러 분야에서 이루어지고 있다. 최근에는 베지어 곡선을 이용한 자율 주행 자동차의 경로 탐색에 대한 연구가 있었다. [1] 또 베지어 곡선은 흔히 말하는 부드러운 곡선을 만드는 방법을 제시하여 글자 폰트를 만들거나 애니메이션에서 물체의 움직임을 나타내는 데에 사용된다. [2] 이런 베지어 곡선을 이용한 선행 연구에서는 고차 베지어 곡선을 이용한 도형의 근사가 주를 이루었다. [3, 4] 베지어 곡면의 경우에는 3D 프린터, 홀로그램 및 VR과 AR 등이 있는데 이러한 기술들이 실제로 상용화 되기 위해서는 더 효율적으로 곡면을 다루기 위한 컴퓨터 그래픽 기술이 필요한데 특히 메모리로 인한 처리 시간을 줄이는 기술이 필수적이다. 현재 3D 모델은 저장, 프로그램 사용 등 많은 부분에서 대부분 obj파일 형식으로 사용된다. obj파일에서 곡면을 나타내는 부분을 보면 곡면을 표현하기 위해 삼각형 혹은 사각형 분할을 이용한다는 것을 알 수 있다. 이러한 다각형 분할은 많은 장점이 있지만, 베지어 곡면을 이용한 압축에 비해 구면과 같은 부드러운 곡면을 표현하는 능력이 떨어진다는 단점도 있다. 고차 베지어 곡선, 곡면을 사용하는 선행연구와 달리 본 연구에서는 모델들을 근사할때 베지어 곡선을 2차 베지어 곡선, (2,2) 차 베지어 곡면만으로 제한하였다. 그리고 이렇게 제한하므로서 얻는 특징들을 활용하여 곡선을 근사하여 저장하는데 필요한 메모리 용량을 줄일 수 있도록 하였다. 이 과정에서 obj 파일 형식으로 주어진 3차원 공간상의 곡면을 조각별로 분할하여 베지어 곡면을 통해 근사 및 압축하는 방법을 제시하였다. 베지어 곡선, 곡면에 움직임을 주는 방법을 3차원으로 확장하여 3차원 곡선, 곡면의 움직임을 줄 수 있음을 보였다. [5] 이것은 베지어 곡면 곡선의 고유한 특성 때문에 메모리의 측면에서의 이점을 가진다. 본 연구에서는 Section II.2의 후속연구로써 모델 모형 뿐만이 아닌 색상 저장 기법을 개선하는 방법을 제시하고자 한다. 이 방법을 활용하면 기존의 mtl저장 기법에 있는 색깔 반사도 저장 형식을 개선할 수도 있다.

그리고 색상에 대한 저장형식도 개선함으로써 메모리를 압축하는 효과를 기대할 수 있다.

I.2 Theorectical background

I.2.1 베지어 곡선과 곡면, 가중치 베지어 곡선과 곡면

Defintion I.1. *n*차 베지어 다항식(Bézier polynomial)은 $n+1$ 개의 점 $\mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n$ 에 대해 다음과 같이 주어진다.

$$\mathbf{B} : [0, 1] \rightarrow \mathbb{R}^2, \mathbf{B}(t) = \sum_{i=0}^n \binom{n}{i} (1-t)^{n-i} t^i \mathbf{P}_i \quad (1)$$

이때 *n*차 베지어 다항식에 의한 폐구간 $[0, 1]$ 의상을 *n*차 베지어 곡선(Bézier curve)이라 한다. 또한 점 $\mathbf{P}_0, \dots, \mathbf{P}_n$ 을 이 베지어 곡선의 조절점(control point)이라 한다.

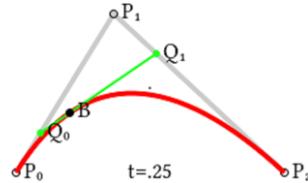


Figure 1. 2차 베지어 곡선

곡선의 조절점을 $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$ 라고 할 때 식은 $\mathbf{B}(t) = (1-t)^2 \mathbf{P}_0 + 2t(1-t) \mathbf{P}_1 + t^2 \mathbf{P}_2$ ($t \in [0, 1]$)이다.

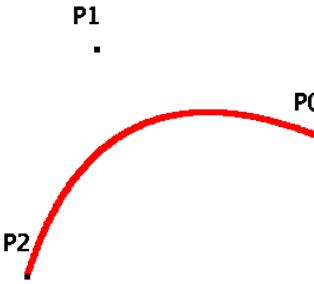
편의상 베지어 곡선을 BC로 표기하자.

Defintion I.2. $(n+1)(m+1)$ 개의 점 \mathbf{b}_{ij} ($i = 0, 1, \dots, n$ and $j = 0, 1, \dots, m$)에 대해 다음 2변수 다항식을 생각하자.

$$\mathbf{x} : [0, 1] \times [0, 1] \rightarrow \mathbb{R}^3, \mathbf{x}(u, v) = \sum_{i=0}^n \sum_{j=0}^m B_i^n(u) B_j^m(v) \mathbf{b}_{ij} \quad (2)$$

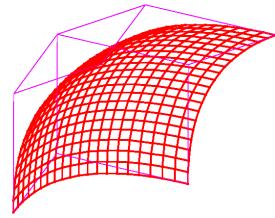
여기서 Bernstein 다항식 $B_i^n(t) = \binom{n}{i} (1-t)^{n-i} t^i$ 로 주어진다. \mathbf{x} 에 의한 단위 정사각형 $[0, 1]^2$ 의상을 (n, m) -차 베지어 곡면(Bézier surface)이라 한다. 마찬가지로 점 \mathbf{b}_{ij} 를 이 베지어 곡면의 조절점이라 한다. [7]

편의상 베지어 곡면을 BS로 표기하자.



(a) 베지어 곡선

Figure 2. 베지어 곡선(a)과 베지어 곡면(b)



(b) 베지어 곡면

Defintion I.3. *n*차 가중치 베지어 다항식(**weighted Bézier polynomial**)은 $n+1$ 개의 점 $\mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n$, $n+1$ 개의 RGB 값을 열상에 배열한 색상 벡터 $\mathbf{Q}_0, \mathbf{Q}_1, \dots, \mathbf{Q}_n$, 위치벡터, 색상벡터를 순서대로 열상에 배열한 색상 조절점 벡터 $\mathbf{K}_0, \mathbf{K}_1, \dots, \mathbf{K}_n$ 에 대해 다음과 같이 주어진다.

$$\begin{aligned}\mathbf{B}(t) &= \sum_{i=0}^n \binom{n}{i} (1-t)^{n-i} t^i \mathbf{P}_i \\ \mathbf{C}(t) &= \sum_{i=0}^n \binom{n}{i} (1-t)^{n-i} t^i \mathbf{Q}_i \\ \mathbf{F}(t) &= \sum_{i=0}^n \binom{n}{i} (1-t)^{n-i} t^i \mathbf{K}_i\end{aligned}\tag{3}$$

이때 *n*차 베지어 다항식에 의한 폐구간 $[0, 1]$ 의 점 $\mathbf{B}(t)$ 에 $\mathbf{C}(t)$ 색상을 입힌 상을 *n*차 가중치 베지어 곡선(**Bézier curve**)이라 한다. 또한 점 $\mathbf{P}_0, \dots, \mathbf{P}_n$ 을 이 가중치 베지어 곡선의 조절점(**control point**)이라 한다. $\mathbf{K}_0, \dots, \mathbf{K}_n$ 을 이 가중치 베지어 곡선의 **RGB 조절점(RGB control point)**이라 한다.

편의상 가중치 베지어 곡선을 WBC로 표기하자.

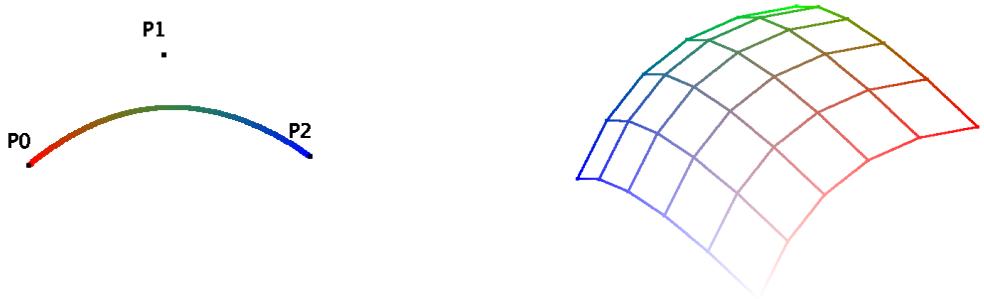
Defintion I.4. $(n+1)(m+1)$ 개의 점 \mathbf{b}_{ij} , $(n+1)(m+1)$ 개의 색상벡터 \mathbf{q}_{ij} ($i = 0, 1, \dots, n$ and $j = 0, 1, \dots, m$) 색상 조절점 벡터 \mathbf{k}_{ij} ($i = 0, 1, \dots, n$ and $j = 0, 1, \dots, m$)에 대해 다음 2변수 다항

식을 생각하자.

$$\begin{aligned}\mathbf{x}(u, v) &= \sum_{i=0}^n \sum_{j=0}^m B_i^n(u) B_j^m(v) \mathbf{b}_{ij} \\ \mathbf{c}(u, v) &= \sum_{i=0}^n \sum_{j=0}^m B_i^n(u) B_j^m(v) \mathbf{q}_{ij} \\ \mathbf{f}(u, v) &= \sum_{i=0}^n \sum_{j=0}^m B_i^n(u) B_j^m(v) \mathbf{k}_{ij}\end{aligned}\quad (4)$$

여기서 Bernstein 다항식 $B_i^n(t) = \binom{n}{i} (1-t)^{n-i} t^i$ 로 주어진다. \mathbf{x} 에 \mathbf{c} 색상을 입힌 단위 정사각형 $[0, 1]^2$ 의 상을 (n, m) -차 가중치 베지어 곡면(**weighted Bézier surface**)이라 한다. 마찬가지로 점 \mathbf{b}_{ij} 를 이 베지어 곡면의 조절점, \mathbf{k}_{ij} 를 이 베지어 곡면의 RGB 조절점이라 한다.

편의상 가중치 베지어 곡면을 WBS로 표기하자.



(a) 가중치 베지어 곡선

Figure 3. 가중치 베지어 곡선(a)과 가중치 베지어 곡면(b)

(b) 가중치 베지어 곡면

I.2.2 근사 기법

Defintion I.5. 뉴턴-랩슨 방법(Newton-Raphson method)은 미분가능한 실수값 함수의 근을 찾는 알고리즘이다. 미분가능한 함수 $f: (a, b) \rightarrow \mathbb{R}$ 과 도함수 f' 에 대해, 현재 $f(x) = 0$ 의 해의 근사치를 x_n 이라 하자. f 의 선형 근사는 다음과 같다.

$$y = f'(x_n)(x - x_n) + f(x_n)$$

다음 근사치 x_{n+1} 은 위 직선의 x 절편으로 주어진다.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad (5)$$

초기값 x_0 부터 시작해 Eq. (5)을 반복해 더 좋은 근사치를 얻을 수 있다. 그러나 초기값의 선택에 따라 $\{x_n\}$ 이 발산할 수 있다. 그리고 f 가 극을 가지지 않는다면 국소 최솟값을 얻는다.

Defintion I.6. 가우스-뉴턴 방법(Gauss-Newton Method)은 뉴턴-랩슨 방법을 다변수 벡터 함수로 확장한 것이다. 함수 $\mathbf{f}: \mathbb{R}^n \rightarrow \mathbb{R}^m$ 에 대해 $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ 의 해의 근사치를 \mathbf{x}_n 이라 하자. 뉴턴-랩슨 방법과 비슷하게 \mathbf{x}_n 에서 \mathbf{f} 의 선형 근사를 구한다.

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\mathbf{x}_n) + \mathbf{J}_{\mathbf{f}}(\mathbf{x}_n)(\mathbf{x} - \mathbf{x}_n)$$

이때 \mathbf{f} 의 야코비 행렬(Jacobian matrix) $\mathbf{J}_{\mathbf{f}}$ 는 다음과 같은 $m \times n$ 행렬이다.

$$\mathbf{J}_{\mathbf{f}} = \begin{pmatrix} \partial f_1 / \partial x_1 & \partial f_1 / \partial x_2 & \cdots & \partial f_1 / \partial x_n \\ \partial f_2 / \partial x_1 & \partial f_2 / \partial x_2 & \cdots & \partial f_2 / \partial x_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial f_m / \partial x_1 & \partial f_m / \partial x_2 & \cdots & \partial f_m / \partial x_n \end{pmatrix}$$

$\mathbf{f}(\mathbf{x}_{n+1}) = \mathbf{0}^\circ$ 라 하면 $\mathbf{J}_{\mathbf{f}}(\mathbf{x}_n)(\mathbf{x}_{n+1} - \mathbf{x}_n) = -\mathbf{f}(\mathbf{x}_n)^\circ$ 이고, 양변에 $\mathbf{J}_{\mathbf{f}}$ 의 의사역행렬(pesudo-inverse)를 곱해 다음 해의 근사치를 얻는다.

$$\mathbf{x}_{n+1} = \mathbf{x}_n - (\mathbf{J}_{\mathbf{f}}(\mathbf{x}_n)^\top \mathbf{J}_{\mathbf{f}}(\mathbf{x}_n))^{-1} \mathbf{J}_{\mathbf{f}}(\mathbf{x}_n)^\top \mathbf{f}(\mathbf{x}_n)$$

실제로 $m = n = 1^\circ$ 라면 이는 뉴턴-랩슨 방법과 같다.

I.2.3 경계 (Boundary)

Defintion I.7. 위상공간 (X, \mathfrak{I}) 의 부분집합 A 의 내부(interior)와 폐포(closure), 경계(boundary)는 각각 다음과 같이 정의된다 [9].

$$A^\circ = \bigcup \{U \subset A \mid U \in \mathfrak{I}\}$$

$$\bar{A} = \bigcap \{C \supset A \mid X - C \in \mathfrak{I}\}$$

$$\partial A = \bar{A} - A^\circ$$

특히 \bar{A} 는 A 의 원소와 A 의 극한점(limit point)로 이루어진다.

I.2.4 볼록집합 (Convex set)

Defintion I.8. \mathbb{R}^n 의 부분집합 C 가 다음 조건을 만족할 때, C 를 볼록 집합(convex set)이라 부른다.

$$\forall \mathbf{x}, \mathbf{y} \in C, \forall t \in [0, 1], (1-t)\mathbf{x} + t\mathbf{y} \in C$$

Lemma I.9. $C \subset \mathbb{R}^n$ 가 볼록집합이면, C 의 폐포(closure) \bar{C} 도 볼록집합이다.

Proof. 임의의 $\mathbf{x}, \mathbf{y} \in \bar{C}$ 와 $t \in [0, 1]$ 에 대해 $(1-t)\mathbf{x} + t\mathbf{y} \in \bar{C}$ 임을 보이자. $\mathbf{x}, \mathbf{y} \in \bar{C}$ 이므로 적당한 수열 $\{\mathbf{x}_n\}, \{\mathbf{y}_n\} \subset C$ 이 존재해 $\mathbf{x}_n \rightarrow \mathbf{x}, \mathbf{y}_n \rightarrow \mathbf{y}$ 를 만족한다. 각 자연수 n 에 대해 \mathbf{x}_n 과 \mathbf{y}_n 은 볼록집합 C 의 원소이므로 $(1-t)\mathbf{x}_n + t\mathbf{y}_n \in C$ 이다. 이제 $n \rightarrow \infty$ 의 극한을 취하면 $(1-t)\mathbf{x} + t\mathbf{y} \in \bar{C}$ 를 얻는다. \square

Theorem I.10. \mathbb{R}^n 의 볼록 집합 C 의 내부(interior)가 공집합이 아닐 때, C 의 경계(boundary) ∂C 는 S^{n-1} 과 위상동형이다. 여기서 S^{n-1} 은 $(n-1)$ 차원 단위 구면으로, $S^{n-1} = \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\| = 1\}$ 이다.

간단히 증명하면, C 의 내부가 공집합이 아니므로 C 를 평행이동하여 $\mathbf{0} \in C^\circ$ 가 되도록 만들 수 있다. 이제 함수 $f: \partial C \rightarrow S^{n-1}$ 을 $f(\mathbf{x}) = \mathbf{x}/\|\mathbf{x}\|$ 로 정의하면 f 는 위상동형사상이 된다.

I.2.5 오차함수

우리는 2D 인지, 3D인지 모델만 근사를 하는건지, 색상까지 근사를 하는건지에 따라 적절한 오차함수를 상황에 따라 고를 필요가 있다. 2가지 오차함수를 정의하여 상황에 따라 이를 그대로 또는 변형해서 사용할 것이다.

Defintion I.11. 거리공간 (M, d) 의 공집합이 아닌 두 부분집합 X, Y 에 대해 하우스도르프 거리는 다음과 같이 주어진다. [9]

$$d_H(X, Y) = \max \left\{ \sup_{x \in X} \inf_{y \in Y} d(x, y), \sup_{y \in Y} \inf_{x \in X} d(x, y) \right\} \quad (6)$$

하우스도르프 거리는 거리공간 상에서 두 집합이 서로 떨어진 정도를 나타낸다. 실제로, $d_H(X, Y) = 0$ 일 필요충분조건은 X 와 Y 의 폐포가 일치하는 것이다.

Defintion I.12. 모델 부분에서 오차함수는 다음과 같이 정의한다. 분할된 각 영역 내의 점을 \mathbf{P}_k ($1 \leq k \leq K$)라 하고, 이에 대응되는 베지어 곡면 위의 점을 $\mathbf{x}(u_k, v_k) = \mathbf{x}_k$ 라 하자. 이때 오차함수 μ 는 다음과 같이 주어진다.

$$\mu = \max_{\text{각 영역}} \max_{1 \leq k \leq K} \|\mathbf{x}_k - \mathbf{P}_k\| \quad (7)$$

I.2.6 obj,mtl파일

obj는 3D 이미지를 저장하는 표준적인 파일이다. obj 파일은 정점 v , 단위 법선벡터 v_n , 텍스쳐 v_t , 면 F 로 이루어진다. 먼저 v, v_n, v_t 의 x, y, z 성분이 차례로 주어진다. 마지막에 F 가 주어지는데, 각각의 F 는 $v/v_n/v_t$ 블록 3개 또는 4개로 이루어진다. 3개면 삼각형 면, 4개면 사각형 면이다.

앞으로 obj파일의 정점 v 를 \mathbf{P}_k 로 나타낸다. 그리고 obj파일로 주어진 곡면은 면 F 들의 합집합을 의미한다.

```
# object heart
```

```
v -5.7868 -2.8897 6.9550
```

```
v -5.8939 -2.7443 6.7745
```

```
...
```

```

v 1.3498 1.7948 1.8726

# 5636 vertices

vn -0.3934 -0.8264 -0.4029

...
vn 0.2663 0.7947 -0.5454

# 5634 vertex normals

vt 0.1020 0.1559 0.0000

...
vt 0.7205 0.0233 0.0000

# 5974 texture coordinates

g Heart      # o [object name] / g [group name]
usemtl Heart # usemtl [material name]
s 1          # s : smooth
f 1/1/1 2/2991/2 3/1583/3 4/2994/4
...
f 5598/1580/5598 5595/5955/5595 5634/2990/5634 5633/5974/5633

```

mtl은 ambient 반사도 K_a , diffuse 반사도 K_d , specular 반사도 K_s 등으로 이루어진다. 모두 0에서 1 사이의 값을 가지며 각각 물체가 원래 가지고 있는 색에 의한 반사도, 국소적인 색에 의한 반사도, 하이라이트를 일으키는 반사도를 의미한다. 본 연구에서는 K_a 및 K_d 만 사용했다.

I.3 연구질문

본 연구에서의 연구 질문은 다음과 같다.

1. 구간을 어떻게 나눌것인가?
2. 모델들이 베지어 곡선, 베지어 곡면을 이용해 점들을 충분히 근사할 수 있다. (i.e. 오차함수가 0으로 수렴한다)
3. obj 등 기존의 곡면을 저장하는 방식보다 더 높은 압축 효율을 가진다.
4. 기존 mtl과 같이 곡면이 지닌 색, 질감 등의 성질도 베지어 곡면의 특징을 이용해 압축하여 저장할 수 있다.
5. 근사하거나 구현하는 과정에서 시간이 obj저장 기법보다 적게 소모된다.
6. 각 방법마다 새로 정의한 오차함수가 적절한가?
7. 색상이 있어도 충분히 근사 가능하다.

II. Main Contents

우리는 어떠한 모델들을 근사하여 메모리를 압축하는 방법들을 다룰 것이다. 일단 기존의 삼각형 분할처럼 베지어 곡면으로도 저장이 가능하다고 보여야한다. 어떠한 모델을 가져다 놓더라도 일단 (2,2)차 베지어 곡면 하나로는 안된다는 건 스파이크 모형만 제시하여도 튀어 나온 부분이 다 근사가 되지 않으므로 알 수 있다. 그러면 우리는 곡면의 차원을 높이거나 그 곡면을 여러개 둘로서 해결할 수 있다. 이 연구에서는 지금까지의 많은 선행연구와 달리 삼각형을 많이 두는 있는 것처럼 (2,2)차 베지어 곡면을 많이 둘 것이다. 이것은 결국 (2,2)차 베지어 곡면을 두는 방법과 하나의 베지어 곡면을 결정하게 되는 영역의 분할하는 방법을 알아야 함을 의미한다. 그리고 이를 근사하면 이게 알맞게 근사되었는지 알 수 있도록 오차함수를 잡아야 하는데 이를 잘 설정해야 잘 근사되었다고 할 수 있을 것이다.

이 연구에서는 2차원 물체와 3차원 물체를 근사해볼 것이다. 그리고 각각에 대해 바로

색상으로 확장하기에는 무리가 있어 단색 모델이 있을 때 이를 근사해보는 것을 먼저 알아볼 것이다.

각각의 근사에 대해서 알아볼 때 분할하는 방법 그리고 분할 했을 때 곡면을 정의해야 하므로 조절점을 결정하는 방법, 그리고 각각에 적절한 오차함수를 제시하고 적절성을 설명할 것이다.

II.1 2D Model Improvement

먼저 우리는 2D 단색 모델에 대한 근사 방법을 알아볼 것이다.

II.1.1 분할 방법

분할하는 방법을 정의하기 위해서는 조각이랑 조각별이라는 개념을 사용할 필요가 있다. 그리고 분할함에 있어서 오차함수도 새로 정의할 필요가 있다.

Defintion II.1. 곡선 C 가 연속함수 $\gamma: [a:b] \rightarrow \mathbb{R}^3$ 의 상 $\gamma([a,b])$ 로 정의된다고 하자. $a = x_0 < x_1 < \dots < x_n = b$ 인 $\{x_i\}_{i=0}^n$ 에 대해 γ 의 정의역을 $[x_{i-1}, x_i]$ 로 축소시킨 n 개의 함수 $\{\gamma_i\}_{i=1}^n$ 로 나누는 것을 조각별로 나눈다고 한다. 또한 각각의 γ_i 에 의한 의 $[x_{i-1}, x_i]$ 상을 곡선의 조각이라 한다.

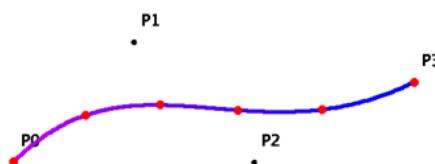


Figure 4. 조각별과 조각
빨간 점을 기준으로 곡선을 조각별로 나누는 그림, 나누어진 각각의 곡선을 조각이라 한다.

Defintion II.2. 곡선 C 를 $\{C_i | i \in I\}$ 와 같이 조각별로 나누었다고 하자. 각 $i \in I$ 에 대해 C_i 를

근사한 곡선을 B_i 라 하면 근사 거리는 다음과 같이 주어진다.

$$\mu_i = d_H(C_i, B_i) \quad (8)$$

전체 곡선 C 와 $B = \cup_{i \in I} B_i$ 사이의 근사 거리는 다음과 같다.

$$\mu = \max_{i \in I} \mu_i \quad (9)$$

Defintion II.3. 곡선 C 와 B 사이의 근사 거리를 μ 라 하자. 임의의 $\varepsilon > 0$ 에 대해 $\mu < \varepsilon$ 이 성립하면 B 는 C 에 충분한 근사가 됐다고 하자.

이제 우리는 어떤 영역 사이의 점이 주어졌을 때 분할하는 방법과 이에 대한 오차함수를 알아보았다. 2D 모델 근사에서의 점은 선적분으로 얻은 길이의 절반이 되는 점으로 한다.

II.1.2 Control Point Determination

어떤 2D 모델을 근사하기 위해서 우리는 분할하는 방법을 다루었다. 그러면 우리는 분할한 영역에 대해서 하나의 베지어 곡선으로 근사시킬 것이기 때문에 조절점 3개를 정해야 한다. 양끝점은 분할한 영역의 끝점으로 하면 우리는 중간 조절점을 찾는 문제만 남는다. 이 조절점을 찾는 과정은 다음 정리를 이용하여 해결한다.

Theorem II.4. 서로 다른 세 점 $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$ 를 조절점으로 가지는 2차 베지어 곡선에 대해 곡선 위의 한 점에서 \mathbf{P}_0 과 \mathbf{P}_2 를 잇는 직선에 내린 수선의 길이가 최대가 되는 점을 \mathbf{Q} 라 하면 $\mathbf{Q} = (\mathbf{P}_0 + 2\mathbf{P}_1 + \mathbf{P}_2)/4$ 이다.

Proof. 우선 2차 베지어 곡선은 다음과 같이 주어진다.

$$\begin{aligned} \mathbf{B}(t) &= (1-t)^2 \mathbf{P}_0 + 2t(1-t) \mathbf{P}_1 + t^2 \mathbf{P}_2 \\ &= (1-t)((1-t)\mathbf{P}_0 + t\mathbf{P}_1) + t((1-t)\mathbf{P}_1 + t\mathbf{P}_2) \end{aligned}$$

세 점 $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$ 는 하나의 평면 α 를 결정한다. $(1-t)\mathbf{P}_0 + t\mathbf{P}_1$ 과 $(1-t)\mathbf{P}_1 + t\mathbf{P}_2$ 는 각각 \mathbf{P}_0 와 $\mathbf{P}_1, \mathbf{P}_2$ 의 $t: (1-t)$ 내분점이므로 α 위에 있다. 또한 $\mathbf{B}(t)$ 도 α 위에 있고, 따라서 2차 베지어 곡선은 하나의 평면에 놓여있다.

이 평면 위에서 다음을 만족하는 직교좌표계 (x', y') 를 만들 수 있다.

$$\mathbf{P}_0 = \begin{pmatrix} -a \\ 0 \end{pmatrix}, \quad \mathbf{P}_1 = \begin{pmatrix} b \\ c \end{pmatrix}, \quad \mathbf{P}_2 = \begin{pmatrix} a \\ 0 \end{pmatrix}, \quad (a, b, c > 0)$$

그러면 \mathbf{Q} 는 y' 좌표가 최대인 점이다. $\mathbf{B}(t)$ 의 y' 좌표는 $2ct(1-t)$ 이므로, $t = 1/2$ 일 때 최대이다. $\mathbf{Q} = \mathbf{B}(1/2) = (b/2, c/2)^\top$ 이므로 위 정리가 성립한다. \square

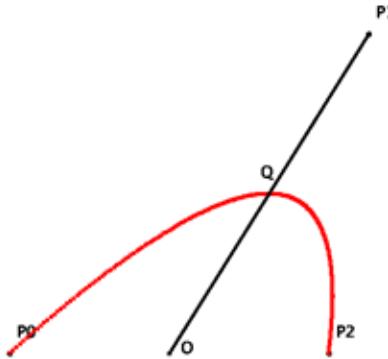


Figure 5. 2차 베지어 곡선의 성질

베지어 곡선 위의 한 점에서 \mathbf{P}_0 과 \mathbf{P}_2 를 잇는 직선에 내린 수선의 길이가 최대가 되는 점을 \mathbf{Q} 라 하면 $\mathbf{Q} = (\mathbf{P}_0 + 2\mathbf{P}_1 + \mathbf{P}_2)/4$ 로 주어진다.

이 연구에서는 위의 방법으로 중간 조절점을 잡는다.

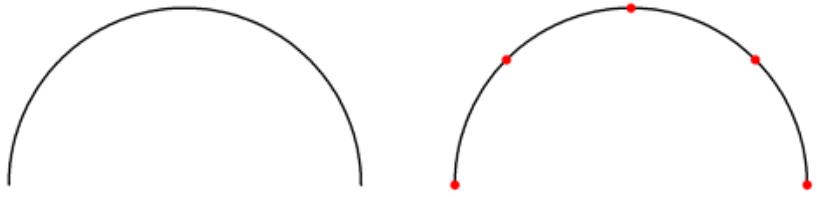
II.1.3 Algorithm

지금까지 분할방법과 중간 조절점을 정하는 방법으로 실제로 원호를 근사해볼 것이다. 위에서 우리는 오차함수를 새로 정의하였는데 어떤 기준값을 설정하여 그 기준값보다 오차함수가 작으면 그대로 종료 아니면 위에서 소개한 분할 방법인 선적분한 길이의 중간 점을

잡아 구간을 분할하고 각각의 조각을 베지어 곡선으로 근사시킬 것이다. 그리고 오차함수가 기준값에서 작아질때까지 계속 반복할 것이다.

일단 조각별로 나누는 과정을 시각적으로 확인해보자 연속함수 $\gamma: [0, 1] \rightarrow \mathbb{R}^3$ 에 대해 $C = \gamma([0, 1])$ 이라 하자. $n \in \mathbb{N}$ 에 대해 곡선을 다음과 같이 2^n 개의 조각으로 나눈다.

$$\begin{aligned}\gamma_i: [0, 1] &\rightarrow \mathbb{R}^3, \gamma_i(t) = \gamma\left(\frac{i+t}{2^n}\right) \\ C_i &= \gamma_i([0, 1]) \quad (i = 0, 1, \dots, 2^n - 1)\end{aligned}\tag{10}$$



(a) $n = 1$

(b) $n = 2$

Figure 6. 곡선을 조각별로 나누기

$n = 2$ 일 때, $\gamma(t) = (\cos \pi t, \sin \pi t, 0)$ 으로 매개되는 단위 반원을 2^n 개의 조각으로 나누기 전(왼쪽)과 후(오른쪽)

이제 각 조각을 근사할 것이다. C_i 위의 한 점에서 $\gamma_i(0)$ 과 $\gamma_i(1)$ 에 내린 수선의 길이가 최대가 되는 점을 중간점이라 정의하고, $\gamma_i(t_{1/2})$ 라 하자. 세 점 $\mathbf{P}_{i,0}, \mathbf{P}_{i,1}, \mathbf{P}_{i,2}$ 를 다음과 같이 정의한다.

$$\begin{aligned}\mathbf{P}_{i,0} &= \gamma_i(0), \\ \mathbf{P}_{i,1} &= 2\gamma_i(t_{1/2}) - \frac{\gamma_i(0) + \gamma_i(1)}{2}, \\ \mathbf{P}_{i,2} &= \gamma_i(1)\end{aligned}\tag{11}$$

이제 $\mathbf{P}_{i,0}, \mathbf{P}_{i,1}, \mathbf{P}_{i,2}$ 를 세 조절점으로 가지는 베지어 곡선은 다음과 같이 주어진다.

$$\begin{aligned}\beta_i(t) &= (1-t)^2 \mathbf{P}_{i,0} + 2t(1-t) \mathbf{P}_{i,1} + t^2 \mathbf{P}_{i,2} \\ B_i &= \beta_i([0, 1])\end{aligned}\tag{12}$$

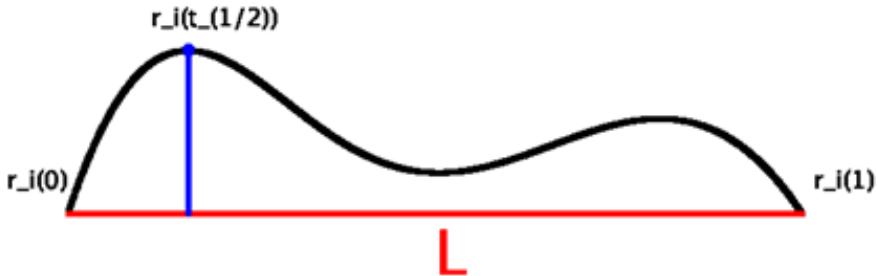


Figure 7. 중간점

조각 $C_i = \gamma_i([0, 1])$ 에서의 중간점은 $\gamma_i(0)$ 과 $\gamma_i(1)$ 을 잇는 직선에 내린 수선의 길이가 최대가 되게 하는 C_i 위의 점을 말하며, $\gamma_i(t_{1/2})$ 로 나타낸다.

Fig. 9를 보면 단위 반원을 근사해본 결과 다음 그림과 같은 수치를 알 수 있었고 이를 그래프로 나타낼 수 있었다. 단위 반원의 경우 $n = 2$ 이면 $\log \mu$ 는 -3.08랑 비슷한 결과가 나타났다.

II.1.4 수렴성 증명

Theorem II.5. 서로 다른 세 점 $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$ 를 조절점으로 가지는 베지어 곡선에 대해 \mathbf{Q} 를 (II.4)와 같이 정의하자. 베지어 곡선 위의 두 점을 잇는 벡터의 \mathbf{P}_0 와 \mathbf{P}_2 를 잇는 직선에 평행한 성분의 최댓값은 $2 \max(\|\mathbf{P}_0 - \mathbf{Q}\|, \|\mathbf{P}_2 - \mathbf{Q}\|)$ 보다 작거나 같다.

Proof. 위의 증명과 마찬가지로 2차 베지어 곡선을 포함하는 평면에서 u, v 축을 설정하자. 일반성을 잃지 않고 $a, b, c > 0$ 이라 할 수 있다. 다음 두 가지 경우가 가능하다. 첫 번째 경우 자명하게 최댓값은 $2a$ 이다.

$$2a \leq 2\sqrt{\left(a + \frac{b}{2}\right)^2 + \left(\frac{c}{2}\right)^2} = \|\mathbf{P}_0 - \mathbf{Q}\| \quad (13)$$

두 번째 경우 베지어 곡선 위의 한 점은 다음과 같이 나타낼 수 있다.

$$\begin{aligned} \mathbf{B}(t) &= (1-t)^2 \mathbf{P}_0 + 2t(1-t) \mathbf{P}_1 + t^2 \mathbf{P}_2 \\ &= (-a(1-t)^2 + 2bt(1-t) + at^2, 2ct(1-t)) \end{aligned} \quad (14)$$

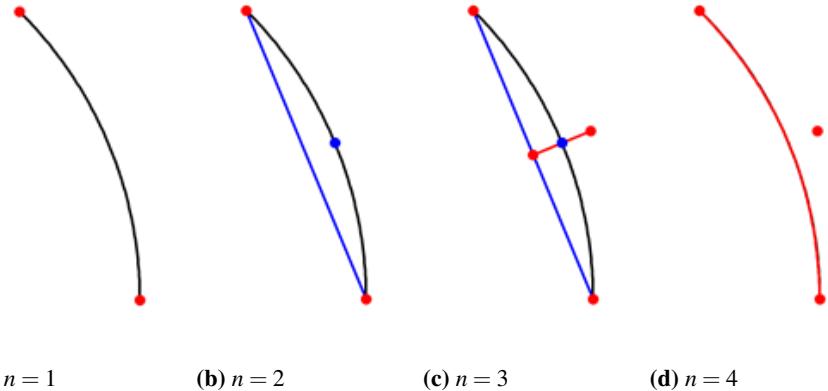


Figure 8. 각 조각을 근사하기

조각 $\gamma_i([0, 1])$, $\gamma_i(0) = (P)_{i,0}$ 과 $\gamma_i(1) = (P)_{i,2}$ 을 잇는 직선과 중간점 $\gamma_i(t_{1/2})$, 중간점을 이용해 얻은 베지어 곡선의 조절점 $\mathbf{P}_{i,1} = 2\gamma_i(t_{1/2}) - \frac{\gamma_i(0) + \gamma_i(1)}{2}$, 베지어 곡선의 세 조절점 $\mathbf{P}_{i,0}, \mathbf{P}_{i,1}, \mathbf{P}_{i,2}$ 를 이용해 그린 2차 베지어 곡선.

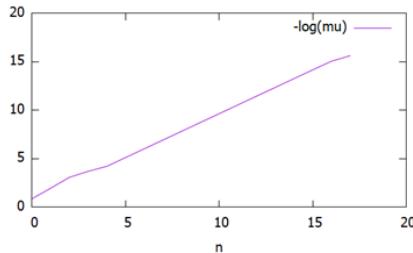


Figure 9. 근사수치

조각 근사 거리: 단위 반원을 근사할 때, n 의 값에 대한 $-\log \mu$ 의 그래프. 이 증가함에 따라 μ 는 기하급수적으로 감소하는 경향을 보인다.

베지어 곡선 위의 두 점을 잇는 벡터의 u 성분은 다음 식의 값보다 작거나 같다.

$$\begin{aligned}
 & (-a(1-t)^2 + 2bt(1-t) + at^2) - (-a) \\
 &= -2bt^2 + 2(a+b)t \\
 &= -2b \left(t - \frac{a+b}{2b} \right)^2 + \frac{(a+b)^2}{2b}
 \end{aligned} \tag{15}$$

베지어 곡선의 개형이 위와 같으므로 이 이차함수는 $t \in [0, 1]$ 에서 최댓값을 가진다. 따라

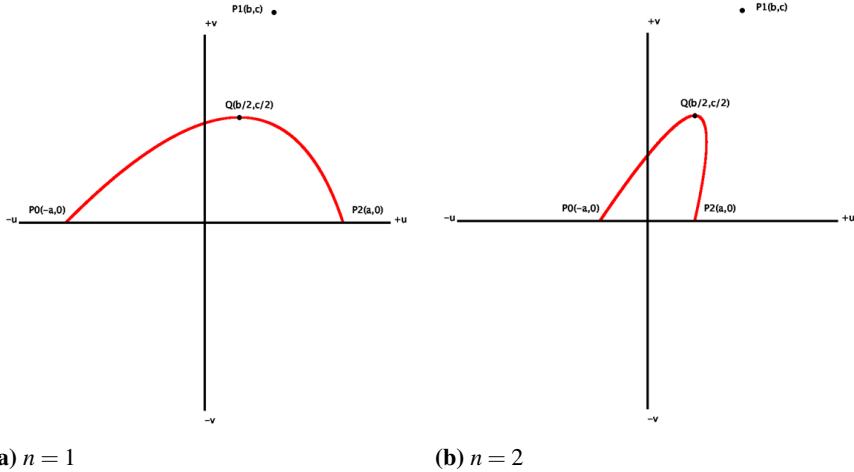


Figure 10. 베지어 곡선의 매개화

베지어 곡선 위의 모든 점은 한 평면상에 존재하기 때문에 축과 축을 설정할 수 있다. 특히, 으로 놓을 수 있다. 일반성을 잃지 않고 이라 할 경우 위 두 가지 경우가 가능하다.

서 $(a+b)/2b \leq 1$ 이 성립하고 이는 $a \leq b$ 와 같다.

$$\begin{aligned} \frac{(a+b)^2}{2b} &= \frac{a^2}{2b} + a + \frac{b}{2} \leq \frac{b}{2} + a + \frac{b}{2} = a + b \\ 2 \| \mathbf{P}_0 - \mathbf{Q} \| &= 2 \sqrt{\left(a + \frac{b}{2}\right)^2 + \left(\frac{c}{2}\right)^2} \geq 2a + b \end{aligned} \quad (16)$$

따라서

$$\frac{(a+b)^2}{2b} \leq a+b \leq 2a+b \leq 2 \| \mathbf{P}_0 - \mathbf{Q} \| \quad (17)$$

이제 지금까지 소개했던 성질들을 이용하여 충분한 근사 가능성에 관한 것을 증명할 것이다. 임의의 $\varepsilon > 0$ 에 대해 $\mu < \varepsilon$ 이 성립하는 n 이 존재한다는 것은 $\lim_{n \rightarrow \infty} \mu = 0$ 의 필요조건이다. 즉, 충분한 근사 가능성에 대한 연구의 방향은 $\lim_{n \rightarrow \infty} \mu = 0$ 을 증명하는 것을 목표로 한다. μ 의 정의에 따라 모든 $i = 0, 1, \dots, 2^n - 1$ 에 대해 $\lim_{n \rightarrow \infty} \mu_i = 0$ 을 보이는 것과 같다.

$\mu_i = d_H(C_i, B_i)$ 는 계산하기 어렵지만 $d_H(C_i, L)$ 과 $d_H(B_i, L)$ 은 할 수 있다. 여기서 L 은 조각의 양 끝점을 잇는 선분이다. 직관적으로 n 이 충분히 크면 C 와 B 가 연속함수의 상이므로

$d_H(C_i, L)$ 과 $d_H(B_i, L)$ 이 모두 0으로 수렴할 것으로 예상된다. 그래서 연속함수의 정의인 $\varepsilon - \delta$ 논법을 이용하여 이를 증명한다.

이 근사거리를 통한 오차 분석의 신뢰성은 높다. 여기서 근사 거리는 원래 곡선과 근사한 곡선 사이의 하우스도르프 거리로 정의한다. 하우스도르프 거리는 거리공간 (\mathbf{M}, d) 의 두 부분집합이 얼마나 멀리 있는가를 나타낸다. $d_H(\mathbf{X}, \mathbf{Y}) = 0$ 일 필요충분조건은 \mathbf{X} 와 \mathbf{Y} 가 동일한 폐포를 가지는 것으로, \mathbf{X}, \mathbf{Y} 가 닫힌 집합일 때는 $\mathbf{X} = \mathbf{Y}$ 와 동치이다. 따라서 하우스도르프 거리는 본 연구에서 두 곡선 사이의 오차를 나타내기 충분하다. 베지어 곡선으로 원호를 근사하는 선행 연구에서도 하우스도르프 거리를 사용했으므로 그 신뢰성도 높다. [3]

Theorem II.6. 임의의 곡선 C 와 위 방법에 따라 얻은 B 사이의 근사거리 μ 에 대해 다음이 성립한다.

$$\lim_{n \rightarrow \infty} \mu = 0 \quad (18)$$

Lemma II.7. 거리공간 (M, d) 의 공집합이 아닌 유계인 세 닫힌 부분집합 X, Y, Z 에 대해 다음이 성립한다.

$$d_H(X, Y) \leq d_H(X, Z) + d_H(Z, Y) \quad (19)$$

Proof. 먼저 $d(x, Y) = \inf_{y \in Y} d(x, y)$ 과 $d(X, Y) = \sup_{x \in X} d(x, Y)$ 에 대해

$$d(x, Y) \leq d(x, Z) + d(Z, Y) \quad (20)$$

가 성립함을 보이자

$$\begin{aligned} d(x, Z) &= \inf_{z \in Z} d(x, z) = \min_{z \in Z} d(x, z) = d(x, z_0) \quad (z_0 \in Z) \\ d(z_0, Y) &= \inf_{y \in Y} d(z_0, y) = \min_{y \in Y} d(z_0, y) = d(z_0, y_0) \quad (y_0 \in Y) \end{aligned} \quad (21)$$

따라서

$$\begin{aligned}
 d(x, Y) &\leq d(x, y_0) \\
 &\leq d(x, z_0) + d(z_0, y_0) \\
 &= d(x, Z) + d(z_0, Y) \\
 &\leq d(x, Z) + d(Z, Y)
 \end{aligned} \tag{22}$$

$x \in X$ 에 대해서

$$\begin{aligned}
 d(x, Y) &\leq d(x, Z) + d(Z, Y) \\
 &\leq d(X, Z) + d(Z, Y) \\
 &\leq \max(d(X, Z), d(Z, X)) + \max(d(Z, Y), d(Y, Z)) \\
 &= d_H(X, Z) + d_H(Z, Y)
 \end{aligned} \tag{23}$$

$$d(X, Y) \leq \sup_{x \in X} d(x, Y) \leq d_H(X, Z) + d_H(Z, Y) \tag{24}$$

비슷하게 $d(Y, X) \leq d_H(X, Z) + d_H(Z, Y)$ 를 얻을 수 있다.

$$d_H(X, Y) = \max(d(X, Y), d(Y, X)) \leq d_H(X, Z) + d_H(Z, Y) \tag{25}$$

□

○]제 Theorem II.6에 대한 증명은 다음과 같다.

Proof. 각 $i = 0, 1, \dots, 2^n - 1$ 에 대해서

L 을 $\gamma_i(0)$ 와 $\gamma_i(1)$ 을 잇는 선분이라 가정하자.

$\gamma_i(t_{1/2})$ 에서 L 을 포함한 직선에 내린 수선의 길이를 h 라 하면

$$d_H(C_i, L) \leq \sup_{c \in C_i} \sup_{l \in L} \| \mathbf{c} - \mathbf{l} \|$$

$\| \mathbf{c} - \mathbf{l} \|$ 에서 L 에 수직인 성분의 최댓값은 정의에 따라 h 이다. L 에 평행한 성분의 최댓값은 $\max(\max_{t \in [0,1]} \| \gamma_i(t) - \gamma_i(0) \|, \max_{t \in [0,1]} \| \gamma_i(t) - \gamma_i(1) \|)$ 보다 작거나 같다.

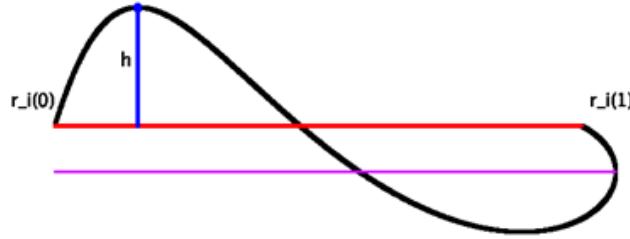


Figure 11. L 에 평행한 성분

$\gamma_i(0)$ 과 $\gamma_i(1)$ 을 잇는 선분 L (빨간색). $\mathbf{c} \in C_i, \mathbf{l} \in L$ 에 대해 $\|\mathbf{c} - \mathbf{l}\|$ 의 L 에 수직인 성분의 최댓값은 h (파란색), L 에 평행한 성분의 최댓값은 보라색 선분의 길이와 같다.

$$\begin{aligned}
 d_H(C_i, L) &\leq \sup_{\mathbf{c} \in C_i} \sup_{\mathbf{l} \in L} \|\mathbf{c} - \mathbf{l}\| \\
 &\leq h + \max \left(\max_{t \in [0,1]} \|\gamma_i(t) - \gamma_i(0)\|, \max_{t \in [0,1]} \|\gamma_i(t) - \gamma_i(1)\| \right) \\
 &\leq 2 \max \left(\max_{t \in [0,1]} \|\gamma_i(t) - \gamma_i(0)\|, \max_{t \in [0,1]} \|\gamma_i(t) - \gamma_i(1)\| \right)
 \end{aligned} \tag{26}$$

중간 조절점을 설정한 과정 Theorem II.5에서 우리는 B_i 위의 한 점에서 L 을 포함한 직선에 내린 수선의 길이의 최댓값도 h 인것을 알 수 있다.

$$\begin{aligned}
 d_H(B_i, L) &\leq \sup_{\mathbf{b} \in B_i} \sup_{\mathbf{l} \in L} \|\mathbf{b} - \mathbf{l}\| \\
 &\leq h + 2 \max (\|\gamma_i(t_{1/2}) - \gamma_i(0)\|, \|\gamma_i(t_{1/2}) - \gamma_i(1)\|) \\
 &\leq 3 \max (\|\gamma_i(t_{1/2}) - \gamma_i(0)\|, \|\gamma_i(t_{1/2}) - \gamma_i(1)\|) \\
 &\leq 3 \max \left(\max_{t \in [0,1]} \|\gamma_i(t) - \gamma_i(0)\|, \max_{t \in [0,1]} \|\gamma_i(t) - \gamma_i(1)\| \right)
 \end{aligned} \tag{27}$$

따라서

$$\begin{aligned}
 d_H(C_i, B_i) &\leq d_H(C_i, L) + d_H(C_i, L) \\
 &\leq 5 \max \left(\max_{t \in [0,1]} \|\gamma_i(t) - \gamma_i(0)\|, \max_{t \in [0,1]} \|\gamma_i(t) - \gamma_i(1)\| \right)
 \end{aligned} \tag{28}$$

$\gamma(t)$ 가 연속이므로 임의의 $\varepsilon > 0$ 에 대해 다음을 만족하는 $\delta_{i,0}, \delta_{i,1} > 0$ 이 존재한다.

$$\begin{aligned} \left| t - \frac{i}{2^n} \right| < \delta_{i,0} &\implies \left\| \gamma(t) - \gamma\left(\frac{i}{2^n}\right) \right\| < \frac{\varepsilon}{5} \\ \left| t - \frac{i+1}{2^n} \right| < \delta_{i,1} &\implies \left\| \gamma(t) - \gamma\left(\frac{i+1}{2^n}\right) \right\| < \frac{\varepsilon}{5} \end{aligned} \quad (29)$$

○ 이제 N_i 를 $2^{N_i} > \max(1/\delta_{i,0}, 1/\delta_{i,1})$ 를 만족하는 최소의 양의 정수로 정의하자

$n > N_i$ 에 대해

$$1/2^n < \delta_{i,0} \text{이므로 } t \in [0, 1] \text{에 대해 } \left| \frac{i+t}{2^n} - \frac{i}{2^n} \right| < \delta_{i,0}$$

$$\left\| \gamma_i(t) - \gamma_i(0) \right\| = \left\| \gamma\left(\frac{i+t}{2^n}\right) - \gamma\left(\frac{i}{2^n}\right) \right\| < \frac{\varepsilon}{5} \quad (30)$$

마찬가지로 $\left\| \gamma_i(t) - \gamma_i(1) \right\| < \frac{\varepsilon}{5}$ 를 얻을 수 있다.

따라서 $n > N_i$ 면 $\mu_i = d_H(C_i, B_i) < \varepsilon$ 이다.

$$N = \max_{i=0}^{2^n-1} N_i \text{라 정의하면 } n > N \implies \max_{i=0}^{2^n-1} \mu_i < \varepsilon$$

$$\therefore \lim_{n \rightarrow \infty} \mu = 0 \quad (31)$$

□

이 증명을 통해 우리는 근사 거리는 항상 분할하는 횟수가 늘어감에 따라 0으로 수렴함을 알 수 있었다. 그러므로 우리는 임의의 연속된 곡선에 대해 충분한 근사가 항상 가능하다고 결론 지을 수 있다.

II.2 3D model improvement

이제 2D 모델 근사에 대한 방법을 알아보았으니 3D 모델도 비슷하게 근사가 가능하다는 것을 보여줄 것이다.

II.2.1 분할 방법

주어진 모델을 하나의 베지어 곡면으로 나타내려면 오차가 너무 커진다. 그래서 우리는 obj 파일의 정점 혹은 point cloud를 분할하고, 분할된 각 영역의 곡면을 하나의 베지어 곡면으로 근사하는 방법을 택했다. 이를 위해 고안한 두 가지 분할 방법이 있다.

- **8진 트리.** 8진 트리는 카테시안 좌표계를 기준으로 하며, 원점(기준점)에 대해 공간을 xy 평면, yz 평면, zx 평면으로 잘라 8개의 영역으로 나누는 방식이다. 8개 영역으로 분할되기 때문에 8개의 자식 노드가 생긴다고 볼 수 있다. 8진 트리 방식의 장점은 모델의 제한 없이 항상 적용 가능하다는 점이다. 그러나 분할된 조각을 베지어 곡면으로 근사하기 어렵고, 조각이 8배씩 많아지므로 시간이 오래 걸린다.
- **원통형 분할.** 원통형 분할은 원통형 좌표계를 기준으로 하며, (r, θ, z) 로 표현되는 정점을 θ 와 z 에 따라 나눈다. $\theta - z$ 평면에 각 정점을 표현하고 4^n 등분하여 영역을 분할하는 방식이다. 원통형 분할은 같은 (θ, z) 값에 대해 두 개 이상의 점이 존재하면 적용할 수 없다는 단점이 있다. 그러나 8진 트리 방법보다 시간이 적게 소모되고, 최종적으로 만들어진 곡면을 조각적으로 정의된 연속함수 $\mathbf{x}^*: \mathbb{R}^2 \rightarrow \mathbb{R}^3$ 에 대해 $\mathbf{x}^*([0, 1]^2)$ 으로 표현할 수 있다는 장점이 있다. 이 점은 모델의 RGB 색을 압축할 때도 사용될 수 있다.

우리는 원통형 분할을 이용하기로 결정했다. 또한 위에서 설명한 단점을 없애기 위해 연구 대상을 축소했다. \mathbb{R}^3 의 볼록 집합의 경계로 표현되는 곡면만을 근사한다. Theorem I.10에 따라 이는 2차원 곡면이 된다.

원통형 분할을 하기 앞서 몇 가지 사전 작업이 필요하다. 정점 \mathbf{P}_k 중 가장 거리가 먼 두 정점을 z 축 위에 올리는 일이다. 원점이 모델 밖에 있으면 같은 (θ, z) 에 대해 두 개 이상의 점이 존재할 수 있다. 또는 기울어진 타원체를 생각하면, 거리가 가장 먼 두 점이 z 축 위에 있는 게 좋다는 걸 알 수 있다. 또한 원기둥의 밑면과 같이 z 축에 수직인 면이 문제가 될 수

있지만, 이 작업을 통해 이런 문제를 해결할 수 있다.

가장 거리가 먼 두 점을 $(x_1, y_1, z_1), (x_2, y_2, z_2)$ 라 하자. 각 정점을 평행이동하여 두 점의 중심이 원점에 오게 한다. 아래의 과정은 모두 우변의 항을 좌변에 대입한다고 할 때 성립하는 식이다.

$$\mathbf{P}_k = \mathbf{P}_k - \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}, \frac{z_1 + z_2}{2} \right)^\top$$

이제 xy 회전, yz 회전을 통해 두 점을 z 축 위로 올린다. 각각 다음과 같다.

$$\begin{aligned}\mathbf{P}_k &= \begin{pmatrix} x_1 & y_1 & 0 \\ -y_1 & x_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{P}_k \\ \mathbf{P}_k &= \begin{pmatrix} z_1 & 0 & -x_1 \\ 0 & 1 & 0 \\ x_1 & 0 & z_1 \end{pmatrix} \mathbf{P}_k\end{aligned}$$

최종적으로 거리가 가장 먼 두 점의 좌표를 $(0, 0, \pm h)$ 라 한다.

θ 와 z 의 범위는 각각 $0 \leq \theta < 2\pi$, $-h \leq z \leq h$ 이므로, 원통형 분할을 하면 각 영역은 $\theta_i = 2\pi i / 2^n$, $z_j = -h + 2hj / 2^n$ ($0 \leq i, j < 2^n$)을 경계로 한다.

II.2.2 Control Point Determination

분할된 곡면을 근사하는 베지어 곡면을 결정하기 위해서 9개의 조절점이 필요하다. 결정된 베지어 곡면이 연속적으로 이어지기 위해서는 \mathbf{b}_{11} 를 제외한 나머지 조절점은 영역 안에 있는 정점들과는 무관하게 결정되어야 한다. 각 영역의 경계는 $\theta = \theta_i$ 혹은 $z = z_j$ 로 주어지므로, 직선 $\ell: \theta = \theta_i, z = z_j$ 와 obj 파일의 면 F 의 교점으로 네 조절점 $\mathbf{b}_{00}, \mathbf{b}_{02}, \mathbf{b}_{20}, \mathbf{b}_{22}$ 를 구한다. 여기서 직선과 삼각형의 교점을 찾는 빠른 알고리즘인 Möller-Trumbore intersection algorithm 을 사용한다. [8] 우리는 2D model 근사에서 중간 조절점을 잡은 방법을 이용하여 네 조절점 $\mathbf{b}_{01}, \mathbf{b}_{10}, \mathbf{b}_{12}, \mathbf{b}_{21}$ 을 얻을 수 있다. (2,2)차 베지어 곡면에서, $u = 0, u = 1, v = 0, v = 1$ 인 부분은 각각 2차 베지어 곡선이 되기 때문이다. 두 영역의 경계 $\theta = \theta_i, z \in [z_j, z_{j+1}]$ 혹은 F 들의

교선을 하나의 2차 베지어 곡면으로 근사하면 된다. Fig. 12은 \mathbf{b}_{11} 을 제외한 8개의 점을 고른 사진이다.

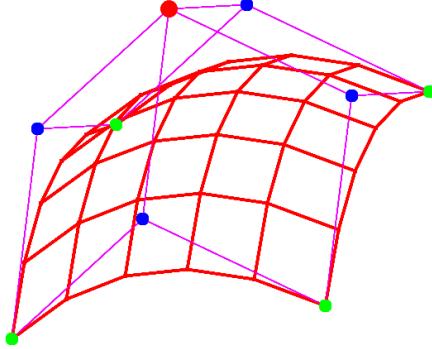


Figure 12. (2,2)-차 베지어 곡면

(2,2)-차 베지어 곡면은 네 모서리가 결정되면 하나의 조절점만 정할 수 있다.

\mathbf{b}_{11} 을 구하기 위해 최소제곱법을 이용한다. 최소제곱법을 이용하는 아이디어는 선행 연구를 참조했다. [10] 최소제곱법을 사용하기 위해서는 obj 파일의 각 정점 \mathbf{P}_k 에 대응되는 베지어 곡면 위의 점 $\mathbf{x}_k = \mathbf{x}(u_k, v_k)$ 을 알아야 한다. 이미 알고 있는 8개 조절점에 대한 항을 \mathbf{y}_k 라 하면 다음과 같이 나타낼 수 있다.

$$\mathbf{x}_k = B_1^2(u_k)B_1^2(v_k)\mathbf{b}_{11} + \mathbf{y}_k$$

이제 $\mathbf{x}_k = \mathbf{P}_k$ ($k = 1, \dots, K$)의 제곱오차 $S = \sum_{k=1}^K \|\mathbf{x}_k - \mathbf{P}_k\|^2$ 를 최소화하기 위해 행렬방정식으로 나타낸다.

$$\begin{pmatrix} B_1^2(u_1)B_1^2(v_1) \\ B_1^2(u_2)B_1^2(v_2) \\ \vdots \\ B_1^2(u_K)B_1^2(v_K) \end{pmatrix} \begin{pmatrix} b_{11}^x & b_{11}^y & b_{11}^z \end{pmatrix} = \begin{pmatrix} P_1^x - y_1^x & P_1^y - y_1^y & P_1^z - y_1^z \\ P_2^x - y_2^x & P_2^y - y_2^y & P_2^z - y_2^z \\ \vdots & \vdots & \vdots \\ P_K^x - y_K^x & P_K^y - y_K^y & P_K^z - y_K^z \end{pmatrix}$$

이때 윗첨자 x, y, z 는 각각 벡터의 x, y, z 성분을 가르킨다. 이 경우에도 제곱오차는 마찬가지로 S 이다. $AX = B$ 꼴의 행렬방정식에서 최소제곱오차를 가지는 \hat{X} 는 $\hat{X} = (A^\top A)^{-1}A^\top B$ 로

주어지므로, \mathbf{b}_{11} 을 다음과 같이 근사할 수 있다.

$$\mathbf{b}_{11} = \frac{\sum_{k=1}^K B_1^2(u_k)B_1^2(v_k)(\mathbf{P}_k - \mathbf{y}_k)}{\sum_{k=1}^K (B_1^2(u_k)B_1^2(v_k))^2} \quad (32)$$

각 k 에 대해 u_k 와 v_k 의 값을 안다면 (32)과 같이 \mathbf{b}_{11} 을 구할 수 있다. 이제 최적의 u_k, v_k 를 찾기 위해 뉴턴-랩슨 방법을 이용한다. 뉴턴-랩슨 방법을 적용할 함수는 $\mathbf{x} - \mathbf{P}_k$ 이다. 즉 현재 $u_{n,k}$ 와 $v_{n,k}$ 가 주어질 때, 다음과 같이 $u_{n+1,k}$ 와 $v_{n+1,k}$ 를 얻는다.

$$u_{n+1,k} = u_{n,k} - \left. \frac{\mathbf{x} - \mathbf{P}_k}{\partial(\mathbf{x} - \mathbf{P}_k)/\partial u} \right|_{u=u_{n,k}} \quad (33)$$

$$v_{n+1,k} = v_{n,k} - \left. \frac{\mathbf{x} - \mathbf{P}_k}{\partial(\mathbf{x} - \mathbf{P}_k)/\partial v} \right|_{v=v_{n,k}} \quad (34)$$

그런데 벡터의 나눗셈은 정의되지 않으므로 이번에도 최소제곱법을 이용한다. 즉 $[\partial(\mathbf{x} - \mathbf{P}_k)/\partial u]\Delta u = \mathbf{x} - \mathbf{P}_k$ 의 제곱 오차를 최소로 하는 스칼라 Δu 를 선택한다. 초기 조건 $u_{0,k} = v_{0,k} = 0.5$ 에 대해 \mathbf{b}_{11} 를 구하고 새로 u_k, v_k 를 구하는 과정을 반복한다. 다음 사진의 과정을 통해 위 과정을 20번 반복하면 u_k 와 v_k 의 값이 거의 일정한 것을 알아냈다.

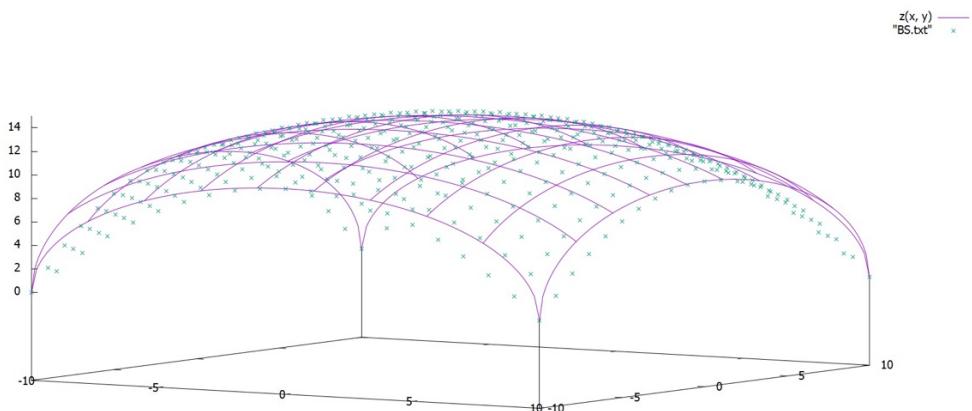


Figure 13. 근사 방법

곡면 $z = \sqrt{200 - x^2 - y^2}$ ($-10 \leq x, y \leq 10$)에 위 근사 방법을 적용한 예시이다. 곡면 위 점의 개수는 $K = 100$ 이며 \mathbf{P}_k 는 x 와 y 의 좌표가 $((2i-9)/10, (2j-9)/10)$ ($i, j = 0, 1, \dots, 9$)인 점이다. 이때 곡면과 근사한 베지어 곡면 사이의 하우스도르프 거리는 약 6.32이다.

II.2.3 Algorithm

본 연구에서는 어떠한 3D 모델이 있을 때 원통형 분할을 이용해 근사를 시작할 것이다. 다음 사진은 원통형 분할을 과정을 시각적으로 보여준 것이다.

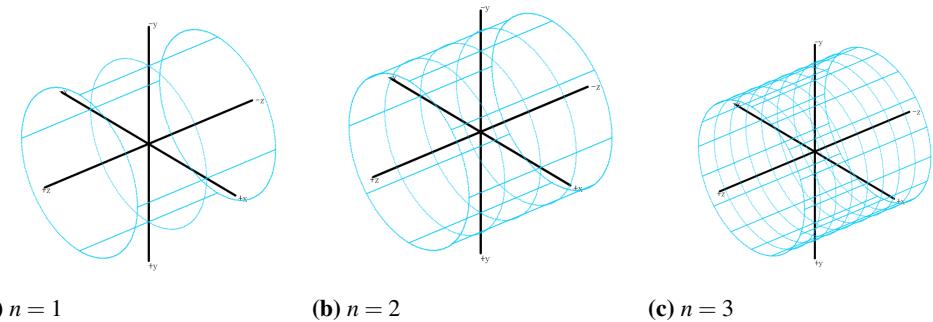


Figure 14. 원통형 분할

원통형 분할은 θ 와 z 의 범위를 이등분하며 이루어진다. 그림에서는 θ 와 z 의 범위를 각각 $2^{n+1}, 2^n$ 등분했다.

분할된 영역의 곡면을 하나의 베지어 곡면으로 근사한다. 각 영역의 베지어 곡면이 연속적으로 이어지기 위해서는 베지어 곡면의 네 모서리($u = 0, u = 1, v = 0, v = 1$)가 영역의 경계에 위치해야 한다. 특히 네 조절점 $\mathbf{b}_{00} = \mathbf{x}(0, 0), \mathbf{b}_{02} = \mathbf{x}(0, 1), \mathbf{b}_{20} = \mathbf{x}(1, 0), \mathbf{b}_{22} = \mathbf{x}(1, 1)$ 은 각각 4개의 영역이 만나는 경계에 위치한다.

분할된 영역이 $\theta_a \leq \theta \leq \theta_{a+1}, z_b \leq z \leq z_{b+1}$ 로 주어진다고 하자. 반직선 $\ell_{ab}: \theta = \theta_a, z = z_b$ 로 정의하고, 네 반직선 $\ell_{ab}, \ell_{a,b+1}, \ell_{a+1,b}, \ell_{a+1,b+1}$ 과 obj파일의 면 F 의 교점을 찾아 각각 $\mathbf{b}_{00}, \mathbf{b}_{02}, \mathbf{b}_{20}, \mathbf{b}_{22}$ 로 둔다. 이때 광선 또는 직선과 삼각형의 교점을 찾는 빠른 알고리즘인 Möller-Trumbore intersection algorithm을 이용한다. [8]

Möller-Trumbore intersection algorithm을 이용하면 영역의 경계 $\theta_a \leq \theta \leq \theta_{a+1}, z = z_b$ 와 F 의 교선을 찾을 수 있다. 식 (2)로 매개화되는 베지어 곡면에서 $u = 0$ 인 부분은 $\mathbf{b}_{00}, \mathbf{b}_{01}, \mathbf{b}_{02}$ 를 세 조절점으로 가지는 2차 베지어 곡선이다. 다음 2차 베지어 곡선의 성질(II.4)을 이용하

면 \mathbf{b}_{01} 을 얻을 수 있다. 영역의 경계 $\theta_a \leq \theta \leq \theta_{a+1}, z = z_b$ 와 ∂C 의 교선을 잡고, 교선 위의 점들 중 \mathbf{b}_{00} 과 \mathbf{b}_{02} 를 잇는 직선에서 가장 멀리 떨어진 점 \mathbf{Q} 를 찾는다. 그러면 $\mathbf{b}_{01} = 2\mathbf{Q} - (\mathbf{b}_{00} + \mathbf{b}_{02})/2$ 로 주어진다. 같은 방법으로 조절점 $\mathbf{b}_{01}, \mathbf{b}_{10}, \mathbf{b}_{12}, \mathbf{b}_{21}$ 을 얻을 수 있고 이를 시각적으로 나타내면 다음과 같다. 이제 조절점 \mathbf{b}_{11} 을 구하기 위해 최소제곱법과 가우스-뉴턴 방법을 이

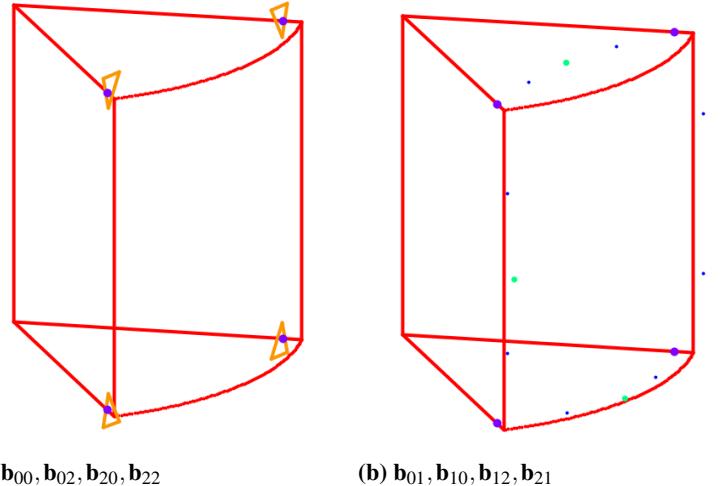


Figure 15. 조절점

\mathbf{b}_{11} 을 제외한 조절점을 얻기 위해 Möller-Trumbore intersecton algorithm을 이용한다. (a) 네 영역이 만나는 (반)직선과 obj파일의 면 F 의 교점으로 $\mathbf{b}_{00}, \mathbf{b}_{02}, \mathbf{b}_{20}, \mathbf{b}_{22}$ 를 얻는다. (b) 영역의 경계와 F 의 교선을 잡고, 2차 베지어 곡선을 성질을 이용해 $\mathbf{b}_{01}, \mathbf{b}_{10}, \mathbf{b}_{12}, \mathbf{b}_{21}$ 를 구한다.

용한다. 이 과정은 선형 연구를 참고했다. [10] 영역 내의 정점을 \mathbf{P}_k ($k = 1, 2, \dots, K$)라 하자. 각 \mathbf{P}_k 에 대응되는 베지어 곡면의 매개변수 u_k 와 v_k 가 주어졌다고 가정한다. $\mathbf{x}_k = \mathbf{x}(u_k, v_k)$ 에서 이미 알고 있는 조절점들에 대한 항을 \mathbf{y}_k 로 두면 다음과 같이 나타날 수 있다.

$$\mathbf{x}_k = \mathbf{x}(u_k, v_k) = B_1^2(u_k)B_1^2(v_k)\mathbf{b}_{11} + \mathbf{y}_k$$

\mathbf{b}_{11} 에 관한 연립방정식 $\mathbf{x}_k = \mathbf{P}_k$ ($1 \leq k \leq K$)는 일반적으로 해를 가지지 않는다. 따라서 제곱오차 $S = \sum_{k=1}^K \|\mathbf{x}_k - \mathbf{P}_k\|^2$ 을 최소로 만드는 \mathbf{b}_{11} 의 값을 구한다. 이를 위해 x, y, z 좌표로

나눠 행렬방정식으로 표현한다.

$$\begin{pmatrix} B_1^2(u_1)B_1^2(v_1) \\ B_1^2(u_2)B_1^2(v_2) \\ \vdots \\ B_1^2(u_K)B_1^2(v_K) \end{pmatrix} \begin{pmatrix} b_{11}^x & b_{11}^y & b_{11}^z \end{pmatrix} = \begin{pmatrix} P_1^x - y_1^x & P_1^y - y_1^y & P_1^z - y_1^z \\ P_2^x - y_2^x & P_2^y - y_2^y & P_2^z - y_2^z \\ \vdots & \vdots & \vdots \\ P_K^x - y_K^x & P_K^y - y_K^y & P_K^z - y_K^z \end{pmatrix}$$

이때 윗첨자 x, y, z 는 각각 벡터의 x, y, z 성분을 나타낸다. 위 행렬방정식의 제곱오차도 마찬가지로 S 임을 알 수 있다. $AX = B$ 꼴의 행렬방정식에서 최소제곱오차를 가지는 $X = (A^\top A)^{-1}A^\top B$ 로 주어지므로, 다음과 같이 \mathbf{b}_{11} 을 구할 수 있다.

$$\mathbf{b}_{11} = \frac{\sum_{k=1}^K B_1^2(u_k)B_1^2(v_k)(\mathbf{P}_k - \mathbf{y}_k)}{\sum_{k=1}^K (B_1^2(u_k)B_1^2(v_k))^2} \quad (35)$$

각 \mathbf{P}_k 에 대해 대응되는 u_k 와 v_k 를 알고 있다면 (35)과 같이 \mathbf{b}_{11} 을 구할 수 있다. 이제 최적의 u_k 와 v_k 를 얻기 위해 가우스-뉴턴 방법을 사용한다. 즉, 각각의 k 에 대해 \mathbf{x}_k 와 \mathbf{P}_k 의 차이를 줄이는 방향으로 u_k 와 v_k 를 갱신한다. 현재 $u_{n,k}$ 와 $v_{n,k}$ 의 값이 $\mathbf{u}_{n,k} = (u_{n,k}, v_{n,k})^\top$ 로 주어질 때, 다음과 같이 $\mathbf{u}_{n+1,k}$ 를 알 수 있다.

$$\mathbf{u}_{n+1,k} = \mathbf{u}_{n,k} - (\mathbf{J}_x^\top \mathbf{J}_x)^{-1} \mathbf{J}_x^\top (\mathbf{x}_k - \mathbf{P}_k) \quad (36)$$

○|때 \mathbf{J}_x 는 $\mathbf{u} = (u, v)^\top$ 에 대한 \mathbf{x} 의 Jacobian matrix이다.

$$\mathbf{J}_x = \begin{pmatrix} \partial x^x / \partial u & \partial x^x / \partial v \\ \partial x^y / \partial u & \partial x^y / \partial v \\ \partial x^z / \partial u & \partial x^z / \partial v \end{pmatrix}$$

앞에서와 마찬가지로 윗첨자는 벡터의 성분을 나타낸다.

초기조건 $u_{0,k} = v_{0,k} = 0.5$ 에 대해, 다음 과정을 반복한다.

1. 식 (35)와 같이 \mathbf{b}_{11} 을 구한다.
2. 식 (36)와 같이 u_k, v_k 를 갱신한다.

실험적으로 약 20번 반복하면 충분했다. 그 이상은 위 과정을 반복해도 Section II.2.2 와 같아 u_k 와 v_k 의 값의 변화가 거의 없었다.

이때, 베지어 곡면은 $0 \leq u, v \leq 1$ 의 범위에서 정의되므로 중간 과정에서 $u_{n,k}$ 또는 $v_{n,k}$ 가 0.01보다 작아지면 0.01로 대체하고, 0.99보다 커지면 0.99로 대체한다. 가우스-뉴턴 방법은 수렴성을 보장할 수 없기 때문에, 이러한 보정은 $u_{n,k}$ 또는 $v_{n,k}$ 가 발산하지 않도록 해준다.

하우스 도르프 거리를 그냥 2D 모델 저장 기법처럼 사용할 수도 있지만 하우스도르프 거리를 곡면에 적용하기에는 문제점이 많다. 기존에는 두 곡선 사이의 하우스도르프 거리를 구하기 위해 구간을 N 등분해 점들 사이의 거리를 구했는데, 이 경우 시간복잡도가 $O(N^2)$ 이 된다. 이번에는 곡선이 아니라 곡면이기에 확인해야 하는 변수가 2배 많아지고, 알고리즘의 시간복잡도가 $O(N^4)$ 으로 제곱이 된다. 그래서 새로운 오차함수를 정의할 필요가 있다.

우리는 Theorem II.6에서 $n \rightarrow \infty$ 일 때 $\mu \rightarrow 0$ 임을 보였다.

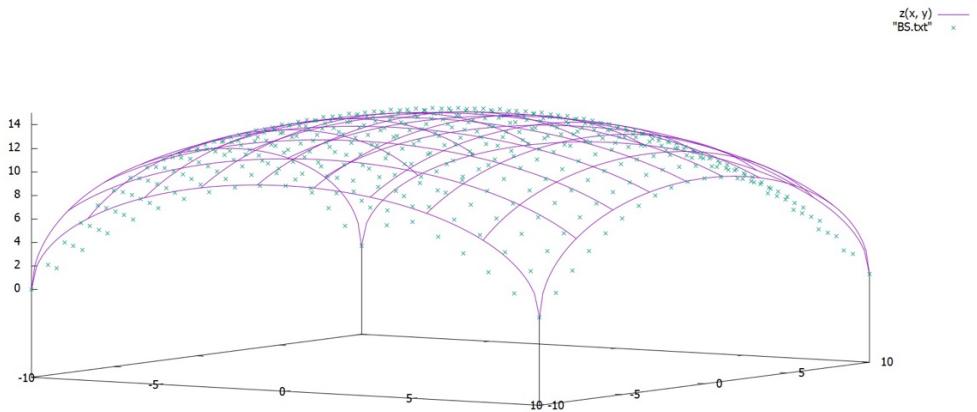


Figure 16. \mathbf{b}_{11} 을 구하는 방법을 적용한 예시

함수 $z = \sqrt{200 - x^2 - y^2}$ 의 그래프에 \mathbf{b}_{11} 을 찾는 방법을 적용한 결과이다. \mathbf{P}_k 는 (x, y) 가 $((2i-9)/10, (2j-9)/10)$ ($i, j = 0, 1, \dots, 9$)인 점이다. 베지어 곡면과 정점들 사이의 하우스도르프 거리는 약 6.32이다.

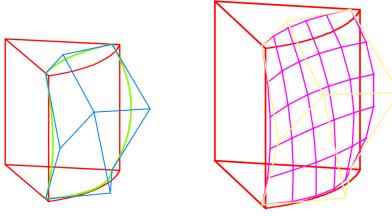


Figure 17. 조절점 - \mathbf{b}_{11}

최소제곱법과 가우스-뉴턴 방법을 통해 \mathbf{b}_{11} 을 구할 수 있다.

II.2.4 수렴성 증명

Theorem II.8. 볼록집합 $C \subset \mathbb{R}^3$ 에 대해 obj파일로 주어진 곡면을 경계 ∂C 라 하자. 다음과 같이 정의된 함수 $\mathbf{f}: [0, 2\pi] \times [-h, h] \rightarrow \partial C$ 는 잘 정의되며, 전사(surjective)이다.

$$\mathbf{f}(\theta_0, z_0) = (\text{경계 } \partial C \text{ 위의 점 중 } \theta = \theta_0, z = z_0 \text{ 인 점})$$

(이때 $r = 0$ 인 점은 어떤 $\theta = \theta_0$ 도 만족한다고 생각한다.)

Proof. 우선 ∂C 는 닫힌 곡면이므로 정의역의 모든 점이 공역의 적어도 하나의 점에 대응된다는 것은 자명하다. \mathbf{f} 가 전사임을 보이기 위해 ∂C 의 모든 점이 $-h \leq z \leq h$ 범위에 있음을 보이면 충분하다. 어떤 점이 $z > h$ 범위에 있다고 가정하자. ∂C 는 \mathbf{P}_k 를 꼭짓점으로 하는 다각형 면들의 합집합이므로, 적어도 하나의 정점 \mathbf{P}_k 가 $z > h$ 범위에 존재해야 한다. 그런데 이는 거리가 가장 먼 두 점이 $(0, 0, \pm h)$ 라는 사실에 모순이다.

$(0, 0, \pm h)^\top$ 은 ∂C 위의 점이므로 \bar{C} 의 원소이다. 보조정리 I.9에 의해 \bar{C} 는 볼록집합이고, 따라서 $\mathbf{0} \in \bar{C}$ 이다. 여기서 우리는 각진 모형이 아닌 부드러운 곡면에 대한 근사에 대해서 다루고 있으므로 $\mathbf{0} \in C^\circ$ 라고 가정할 수 있다. 또한 $\{(0, 0, z)^\top \mid -h < z < h\} \subset C^\circ$ 이다.

$(\theta, z) = (\theta_0, z_0)$ 를 만족하는 서로 다른 두 점이 있다고 가정해 볼 때, $z_0 = \pm h$ 이면 이에 대응되는 점은 하나뿐이므로 $-h < z_0 < h$ 이다. $(0, 0, z_0)$ 는 C 의 내부에 있고, 이를 기준으로 정리 I.10을 사용하면 ϕ 에 의해 두 점이 같은 점으로 사영되므로 모순이 발생한다는 점에서 \mathbf{f} 는 잘 정의되며 전사함수임을 알 수 있다. \square

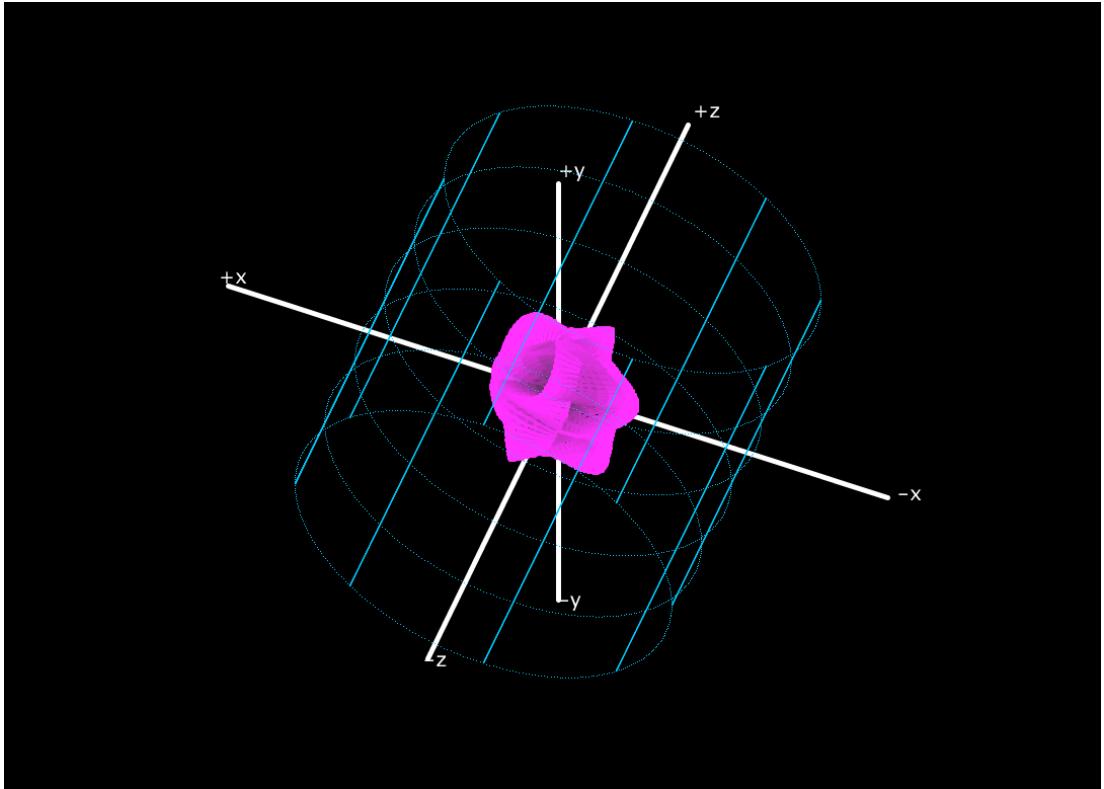


Figure 18. 구면 근사

반지름 64인 구면은 근사한 모델(분홍색)로, Eq. (49)로 정의한 오차함수는 $\mu < 10$ 이다.

또한 ∂C 는 닫힌 곡면이므로 \mathbf{f} 는 연속함수이다. \mathbf{f} 가 연속이 아니라면 ∂C 가 연속적으로 매개화될 수 없고, 곡면이 아니다.

Theorem II.9. 위와 같은 근사 방법에 따라 3D 모델을 근사했다고 하자. 식 (7)로 주어진 오차함수 μ 에 대해 다음이 성립한다.

$$\lim_{n \rightarrow \infty} \mu = 0$$

Proof. $B_i^2(u)B_j^2(v)$ 의 합이 1이므로 다음이 성립한다.

$$\begin{aligned} \|\mathbf{x}_k - \mathbf{P}_k\| &= \left\| \sum_{i,j=0}^2 B_i^2(u_k)B_j^2(v_k)(\mathbf{b}_{ij} - \mathbf{P}_k) \right\| \\ &\leq \sum_{i,j=0}^2 B_i^2(u_k)B_j^2(v_k) \|\mathbf{b}_{ij} - \mathbf{P}_k\| \end{aligned}$$

$\|\mathbf{x}_k - \mathbf{P}_k\|$ 는 $\|\mathbf{b}_{ij} - \mathbf{P}_k\|$ 의 가중평균이므로, $\|\mathbf{b}_{ij} - \mathbf{P}_k\| \rightarrow 0$ 이면 $\|\mathbf{x}_k - \mathbf{P}_k\| \rightarrow 0$ 이다.

정리 II.8의 함수 \mathbf{f} 를 생각하자. $\mathbf{f}: [0, 2\pi] \times [-h, h] \rightarrow \mathbb{R}^3$ 는 콤팩트 집합에서 정의된 연속 함수이므로 Heine-Cantor Theorem에 의해 균등연속(uniformly continuous)이다. 즉, 임의의 $\varepsilon > 0$ 에 대해 다음을 만족하는 $\delta > 0$ 이 존재한다.

$$\sqrt{(\theta - \theta')^2 + (z - z')^2} < \delta \implies \|\mathbf{f}(\theta, z) - \mathbf{f}(\theta', z')\| < \varepsilon$$

제 $\mathbf{P}_k = \mathbf{f}(\theta_k, z_k)$ 라 하고, (θ_k, z_k) 를 포함하는 영역

$$I_{ab} = \left[\frac{2\pi a}{2^n}, \frac{2\pi(a+1)}{2^n} \right] \times \left[-h + \frac{2hb}{2^n}, -h + \frac{2h(b+1)}{2^n} \right]$$

가 (θ_k, z_k) 의 $\delta/2$ -근방에 포함되도록 하는 최소의 자연수 N 을 선택하자. $\mathbf{b}_{00}, \mathbf{b}_{02}, \mathbf{b}_{20}, \mathbf{b}_{22} \in \mathbf{f}(I_{ab})$ 이므로, $n \geq N$ 이면 $\|\mathbf{b}_{00} - \mathbf{P}_k\|, \|\mathbf{b}_{02} - \mathbf{P}_k\|, \|\mathbf{b}_{20} - \mathbf{P}_k\|, \|\mathbf{b}_{22} - \mathbf{P}_k\|$ 는 모두 ε 보다 작다. 또한 $\mathbf{f}(I_{ab})$ 가 \mathbf{b}_{00} 의 ε -근방에 포함되므로 $\|\mathbf{b}_{01} - \mathbf{P}_k\| < \|\mathbf{b}_{01} - \mathbf{Q}\| + \|\mathbf{Q} - \mathbf{P}_k\| < 2\varepsilon$ 성립 한다 (\mathbf{Q} 는 Theorem II.4 참고). 마찬가지로 $\|\mathbf{b}_{10} - \mathbf{P}_k\|, \|\mathbf{b}_{12} - \mathbf{P}_k\|, \|\mathbf{b}_{21} - \mathbf{P}_k\|$ 는 모두 2ε 보다 작다. 따라서 $(i, j) \neq (1, 1)$ 에 대해 $\lim_{n \rightarrow \infty} \|\mathbf{b}_{ij} - \mathbf{P}_k\| = 0$ 이다.

\mathbf{b}_{11} 은 (35)과 같이 주어지고, 이때 \mathbf{y}_k 는 다음과 같다.

$$\mathbf{y}_k = \sum_{(i,j) \neq (1,1)} B_i^2(u_k) B_j^2(v_k) \mathbf{b}_{ij}$$

$\|\mathbf{b}_{11} - \mathbf{P}_k\|$ 는 아래와 같은 부등식으로 계산된다.

$$\begin{aligned} & \|\mathbf{b}_{11} - \mathbf{P}_k\| \\ &= \left\| \frac{\sum_{k=1}^K B_1^2(u_k) B_1^2(v_k) \sum_{(i,j) \neq (1,1)} B_i^2(u_k) B_j^2(v_k) (\mathbf{P}_k - \mathbf{b}_{ij})}{\sum_{k=1}^K (B_1^2(u_k) B_1^2(v_k))^2} \right\| \\ &\leq \frac{\sum_{k=1}^K B_1^2(u_k) B_1^2(v_k) \sum_{(i,j) \neq (1,1)} B_i^2(u_k) B_j^2(v_k) \|\mathbf{P}_k - \mathbf{b}_{ij}\|}{\sum_{k=1}^K (B_1^2(u_k) B_1^2(v_k))^2} \\ &\leq \frac{\sum_{k=1}^K B_1^2(u_k) B_1^2(v_k) (1 - B_1^2(u_k) B_1^2(v_k))}{\sum_{k=1}^K (B_1^2(u_k) B_1^2(v_k))^2} \cdot 2\varepsilon \end{aligned}$$

$$\begin{aligned}
&= \left[\frac{\sum_{k=1}^K B_1^2(u_k)B_1^2(v_k)}{\sum_{k=1}^K (B_1^2(u_k)B_1^2(v_k))^2} - 1 \right] \cdot 2\epsilon \\
&\leq \frac{K}{\sum_{k=1}^K (B_1^2(u_k)B_1^2(v_k))^2} \frac{\epsilon}{2}
\end{aligned}$$

마지막 식에서 다음과 같은 절대부등식을 사용했다.

$$\frac{\sum_{k=1}^K x_k}{\sum_{k=1}^K x_k^2} \leq 1 + \frac{K}{4 \sum_{k=1}^K x_k^2}$$

이는 절대부등식 $x_k \leq x_k^2 + 1/4$ 를 변변이 더하고 $\sum x_k^2$ 으로 나누면 얻을 수 있다.

u_k 와 v_k 가 항상 $[0.01, 0.99]$ 범위에 있으므로 $(B_1^2(u_k)B_1^2(v_k))^2 \geq (2 \times 0.01 \times 0.99)^4 = C$ 이고, 이에 따라 $\|\mathbf{b}_{11} - \mathbf{P}_k\| \leq \epsilon K / 2C$ 이다. 여기서 K 는 영역 안에 있는 점의 개수이다. 전체 점 개수를 K_1 라 한다면 $\|\mathbf{b}_{11} - \mathbf{P}_k\| \leq \epsilon K / 2C \leq \epsilon K_1 / 2C$ 임을 알 수 있다. 모든 $0 \leq i, j \leq 2$ 에 대해 $\lim_{n \rightarrow \infty} \|\mathbf{b}_{ij} - \mathbf{P}_k\| = 0$ 으로, 이들의 가중평균인 $\mathbf{x}_k - \mathbf{P}_k$ 도 $n \rightarrow \infty$ 의 극한에서 0으로 수렴한다.

각 k 에 대해서 $\lim_{n \rightarrow \infty} \|\mathbf{x}_k - \mathbf{P}_k\| = 0$ 으로, 모든 $\epsilon > 0$ 에 대해 적당한 자연수 N_k 가 존재해 $n \geq N_k \implies \|\mathbf{x}_k - \mathbf{P}_k\| < \epsilon$ 을 만족한다. 이제 자연수 N 을 N_k 들의 최댓값으로 정의한다.

$$N = \max_{\text{각 영역}} \max_{1 \leq k \leq K} N_k$$

그러면 모든 $n \geq N$ 에 대해 $\mu < \epsilon$ 으로, $\lim_{n \rightarrow \infty} \mu = 0$ 을 얻는다. \square

정리 II.9는 제시한 근사 방법에 따라 obj파일로 주어진 곡면을 원하는 오차 범위 내에서 베지어 곡면으로 근사할 수 있음을 의미한다.

II.3 2D model RGB implementation

이제 각각 모델이 근사가 된다는 것을 보였으니 실생활에 더 넓게 활용될 수 있도록 RGB 근사도 다룰 것이다.

II.3.1 분할 방법

일단 2D 근사를 해야하는데 2D영역에 각 점에 색깔이 있다. 이를 쉽게 해석하기 위해 본 연구에서는 R,G,B 따로 따로 근사하고 이에 대한 값을 근사해서 더하는 형태로 근사가 진행될 것이다. 그러면 임의의 2D 그림에 대하여 (x,y) 위치에 따라 각각 R,G,B값을 함수로 나타낼 필요가 생긴다.

Defintion II.10. 2D그림에서 (x,y) 위치에 대응되는 R,G,B값을 각각 a,b,c 라고 할때 f_R, f_G, f_B 함수를 각각 **R함수, G함수, B함수**라고 하고 다음과 같이 정의한다.

$$a = f_R(x,y), b = f_G(x,y), c = f_B(x,y) \quad (37)$$

그리고 우리는 이 f_R, f_G, f_B 함수를 베지어 곡면으로 근사시킨 함수를 h_R, h_G, h_B 함수라고 하면 최종적으로 근사한 2D는 3개의 함수를 더하면 나타날 것이다.

지금부터 근사하는 과정은 R함수, G함수, B함수는 똑같이 진행되므로 R함수를 기준으로 설명할 것이다.

여기서 보면 R함수, G함수, B함수는 2개의 독립변수에 의해 값이 나타나지는 양함수이므로 베지어 곡선보다는 베지어 곡면으로 근사하는 것이 옳을 것이다. 그리고 무엇보다 R 함수, G 함수, B 함수의 그래프 개형만 근사하면 되므로 결국 색깔이 없는 베지어 곡면을 근사하는 것과 비슷하게 하면 됨을 의미한다.

그러면 분할하는 방법은 원통형 분할과 비슷하게 다음과 같이 정의한다.

Defintion II.11. 2D RGB 분할은 x,y 하나의 색상을 축으로 하는 3차원 직교좌표계를 기준으로 하며 (x,y) 로 표현되는 점들을 x,y 에 따라 나눈다. xy평면에 각 점들을 표현하고 각 축을 반으로 분할해서 영역을 4^n 등분하여 영역을 분할하는 방식이다.

여기서 x,y 범위는 각각 $0 \leq x \leq a, 0 \leq y \leq b$ 이므로 각 RGB분할을 하면 각 영역은 $x_i = a/2^n, y_i = b/2^n$ 을 경계로 함을 알 수 있다.

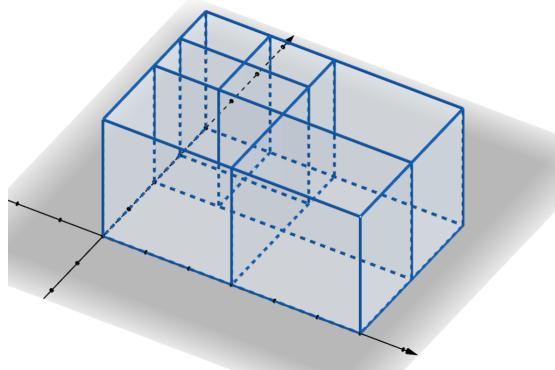


Figure 19. 2D RGB 분할

이 그림에서의 2D RGB 분할은 전반적으로 한번 나누고 그 안에서 오차함수가 기준치보다 높을 때 한번 더 나뉘어진 모습을 보여주고 있다.

II.3.2 Control Point Determination

분할된 영역에서 베지어 곡면을 결정하기 위해서는 9개의 점이 필요함을 생각할 수 있다. 베지어 곡면이 연속적으로 이어지기 위해서는 \mathbf{b}_{11} 를 제외한 나머지 조절점은 영역안에 있는 점들과는 무관하게 결정되어야한다. 각 영역의 경계는 $x = x_i$ 혹은 $y = y_i$ 로 주어지므로 직선 $\ell: x = x_i, y = y_j$ 와 R 함수, G 함수, B 함수의 교점으로 $\mathbf{b}_{00}, \mathbf{b}_{02}, \mathbf{b}_{20}, \mathbf{b}_{22}$ 를 구한다. 여기서 2D 모델에서 모든 점에는 색깔이 있으므로 위에서 쓴 Möller-Trumbore intersection algorithm을 사용할 필요는 없다. 그리고 2D model 근사에서 중간 조절점을 잡은 방법을 이용하여 네 조절점 $\mathbf{b}_{01}, \mathbf{b}_{10}, \mathbf{b}_{12}, \mathbf{b}_{21}$ 을 얻을 수 있다. 이는 (2,2)차 베지어 곡면에서 $u = 0, u = 1, v = 0, v = 1$ 인 부분은 2차 베지어 곡선이 되기 때문이다. 이에 대한 그림은 Fig. 12에 있으므로 생략하겠다.

\mathbf{b}_{11} 를 구하기 위해 최소제곱법을 이용하였다. 최소제곱법을 사용하기 위해서는 R 함수에서 몇개의 데이터 점 \mathbf{P}_k 를 뽑고 이에 대응되는 베지어 곡면위의 점 $\mathbf{x}_k = \mathbf{x}(u_k, v_k)$ 을 알아야 한다. 이미 알고 있는 8개 조절점에 대한 항을 \mathbf{y}_k 라 하면 다음과 같이 나타낼 수 있다.

$$\mathbf{x}_k = B_1^2(u_k)B_1^2(v_k)\mathbf{b}_{11} + \mathbf{y}_k$$

이제 $\mathbf{x}_k = \mathbf{P}_k$ ($k = 1, \dots, K$)의 제곱오차 $S = \sum_{k=1}^K \|\mathbf{x}_k - \mathbf{P}_k\|^2$ 를 최소화하기 위해 행렬방정식

으로 나타낸다.

$$\begin{pmatrix} B_1^2(u_1)B_1^2(v_1) \\ B_1^2(u_2)B_1^2(v_2) \\ \vdots \\ B_1^2(u_K)B_1^2(v_K) \end{pmatrix} \begin{pmatrix} b_{11}^x & b_{11}^y & b_{11}^z \end{pmatrix} = \begin{pmatrix} P_1^x - y_1^x & P_1^y - y_1^y & P_1^z - y_1^z \\ P_2^x - y_2^x & P_2^y - y_2^y & P_2^z - y_2^z \\ \vdots & \vdots & \vdots \\ P_K^x - y_K^x & P_K^y - y_K^y & P_K^z - y_K^z \end{pmatrix}$$

이때 윗첨자 x, y, z 는 각각 벡터의 x, y, z 성분을 가르킨다. 이 경우에도 제곱오차는 마찬가지로 S 이다. $AX = B$ 꼴의 행렬방정식에서 최소제곱오차를 가지는 \hat{X} 는 $\hat{X} = (A^\top A)^{-1}A^\top B$ 로 주어지므로, \mathbf{b}_{11} 을 다음과 같이 근사할 수 있다.

$$\mathbf{b}_{11} = \frac{\sum_{k=1}^K B_1^2(u_k)B_1^2(v_k)(\mathbf{P}_k - \mathbf{y}_k)}{\sum_{k=1}^K (B_1^2(u_k)B_1^2(v_k))^2} \quad (38)$$

각 k 에 대해 u_k 와 v_k 의 값을 안다면 (38)과 같이 \mathbf{b}_{11} 을 구할 수 있다. 이제 최적의 u_k, v_k 를 찾기 위해 뉴턴-랩슨 방법을 이용한다. 뉴턴-랩슨 방법을 적용할 함수는 $\mathbf{x} - \mathbf{P}_k$ 이다. 즉 현재 $u_{n,k}$ 와 $v_{n,k}$ 가 주어질 때, 다음과 같으 $u_{n+1,k}$ 와 $v_{n+1,k}$ 를 얻는다.

$$u_{n+1,k} = u_{n,k} - \left. \frac{\mathbf{x} - \mathbf{P}_k}{\partial(\mathbf{x} - \mathbf{P}_k)/\partial u} \right|_{u=u_{n,k}} \quad (39)$$

$$v_{n+1,k} = v_{n,k} - \left. \frac{\mathbf{x} - \mathbf{P}_k}{\partial(\mathbf{x} - \mathbf{P}_k)/\partial v} \right|_{v=v_{n,k}} \quad (40)$$

그런데 벡터의 나눗셈은 정의되지 않으므로 이번에도 최소제곱법을 이용한다. 즉 $[\partial(\mathbf{x} - \mathbf{P}_k)/\partial u]\Delta u = \mathbf{x} - \mathbf{P}_k$ 의 제곱 오차를 최소로 하는 스칼라 Δu 를 선택한다. 초기 조건 $u_{0,k} = v_{0,k} = 0.5$ 에 대해 \mathbf{b}_{11} 를 구하고 새로 u_k, v_k 를 구하는 과정을 반복한다.

이번에도 u_k 와 v_k 의 값을 거의 일정하게 만들기 위해 위 과정을 20번 반복한다.

II.3.3 Algorithm

우리는 이제 어떻게 하면 근사가 되는지 살펴볼 것이다. 일단 어떠한 R 함수가 있을 때 이를 2D RGB 분할을 할 것이다. 그리고 분할된 영역의 곡면을 하나의 베지어 곡면으로 근사를

해볼 것이다. 분할된 영역의 베지어 곡면이 연속적으로 이어지기 위해서는 베지어 곡면의 네 모서리($u = 0, u = 1, v = 0, v = 1$)가 영역의 경계에 위치해야 한다. 특히 네 조절점 $\mathbf{b}_{00} = \mathbf{x}(0,0), \mathbf{b}_{02} = \mathbf{x}(0,1), \mathbf{b}_{20} = \mathbf{x}(1,0), \mathbf{b}_{22} = \mathbf{x}(1,1)$ 은 각각 4개의 영역이 만나는 경계에 위치한다.

분할된 영역이 $x_a \leq x \leq x_{a+1}, y_b \leq z \leq y_{b+1}$ 로 주어진다고 하자. 직선 $\ell_{ab}: x = x_a, y = y_b$ 로 정의하고, 네 직선 $\ell_{ab}, \ell_{a,b+1}, \ell_{a+1,b}, \ell_{a+1,b+1}$ 과 R함수와의 교점을 찾아 각각 $\mathbf{b}_{00}, \mathbf{b}_{02}, \mathbf{b}_{20}, \mathbf{b}_{22}$ 로 둔다.

그리고 점 (x, y, f_R) 들 중 \mathbf{b}_{00} 과 \mathbf{b}_{02} 를 잇는 직선에서 가장 멀리 떨어진 점 \mathbf{Q} 를 찾는다. 그러면 $\mathbf{b}_{01} = 2\mathbf{Q} - (\mathbf{b}_{00} + \mathbf{b}_{02})/2$ 로 주어진다. 같은 방법으로 조절점 $\mathbf{b}_{01}, \mathbf{b}_{10}, \mathbf{b}_{12}, \mathbf{b}_{21}$ 을 얻을 수 있다.

이제 조절점 \mathbf{b}_{11} 을 구하기 위해 최소제곱법과 가우스-뉴턴 방법을 이용한다. 영역 내의 점 (x, y, f_R) 을 \mathbf{P}_k ($k = 1, 2, \dots, K$)라 하자. 각 \mathbf{P}_k 에 대응되는 베지어 곡면의 매개변수 u_k 와 v_k 가 주어졌다고 가정한다. $\mathbf{x}_k = \mathbf{x}(u_k, v_k)$ 에서 이미 알고 있는 조절점들에 대한 항을 \mathbf{y}_k 로 두면 다음과 같이 나타날 수 있다.

$$\mathbf{x}_k = \mathbf{x}(u_k, v_k) = B_1^2(u_k)B_1^2(v_k)\mathbf{b}_{11} + \mathbf{y}_k$$

\mathbf{b}_{11} 에 관한 연립방정식 $\mathbf{x}_k = \mathbf{P}_k$ ($1 \leq k \leq K$)는 일반적으로 해를 가지지 않는다. 따라서 제곱오차 $S = \sum_{k=1}^K \|\mathbf{x}_k - \mathbf{P}_k\|^2$ 을 최소로 만드는 \mathbf{b}_{11} 의 값을 구한다. 이를 위해 x, y, z 좌표로 나눠 행렬방정식으로 표현한다.

$$\begin{pmatrix} B_1^2(u_1)B_1^2(v_1) \\ B_1^2(u_2)B_1^2(v_2) \\ \vdots \\ B_1^2(u_K)B_1^2(v_K) \end{pmatrix} \begin{pmatrix} b_{11}^x & b_{11}^y & b_{11}^z \end{pmatrix} = \begin{pmatrix} P_1^x - y_1^x & P_1^y - y_1^y & P_1^z - y_1^z \\ P_2^x - y_2^x & P_2^y - y_2^y & P_2^z - y_2^z \\ \vdots & \vdots & \vdots \\ P_K^x - y_K^x & P_K^y - y_K^y & P_K^z - y_K^z \end{pmatrix}$$

이때 윗첨자 x, y, z 는 각각 벡터의 x, y, z 성분을 나타낸다. 위 행렬방정식의 제곱오차도 마찬가지로 S 임을 알 수 있다. $AX = B$ 꼴의 행렬방정식에서 최소제곱오차를 가지는 $X =$

$(A^\top A)^{-1} A^\top B$ 로 주어지므로, 다음과 같이 \mathbf{b}_{11} 을 구할 수 있다.

$$\mathbf{b}_{11} = \frac{\sum_{k=1}^K B_1^2(u_k) B_1^2(v_k) (\mathbf{P}_k - \mathbf{y}_k)}{\sum_{k=1}^K (B_1^2(u_k) B_1^2(v_k))^2} \quad (41)$$

각 \mathbf{P}_k 에 대해 대응되는 u_k 와 v_k 를 알고 있다면 (41)과 같이 \mathbf{b}_{11} 을 구할 수 있다. 이제 최적의 u_k 와 v_k 를 얻기 위해 가우스-뉴턴 방법을 사용한다. 즉, 각각의 k 에 대해 \mathbf{x}_k 와 \mathbf{P}_k 의 차이를 줄이는 방향으로 u_k 와 v_k 를 갱신한다. 현재 $u_{n,k}$ 와 $v_{n,k}$ 의 값이 $\mathbf{u}_{n,k} = (u_{n,k}, v_{n,k})^\top$ 로 주어질 때, 다음과 같이 $\mathbf{u}_{n+1,k}$ 를 알 수 있다.

$$\mathbf{u}_{n+1,k} = \mathbf{u}_{n,k} - (\mathbf{J}_x^\top \mathbf{J}_x)^{-1} \mathbf{J}_x^\top (\mathbf{x}_k - \mathbf{P}_k) \quad (42)$$

이때 \mathbf{J}_x 는 $\mathbf{u} = (u, v)^\top$ 에 대한 \mathbf{x} 의 Jacobian matrix이다.

$$\mathbf{J}_x = \begin{pmatrix} \partial x^x / \partial u & \partial x^x / \partial v \\ \partial x^y / \partial u & \partial x^y / \partial v \\ \partial x^z / \partial u & \partial x^z / \partial v \end{pmatrix}$$

앞에서와 마찬가지로 윗첨자는 벡터의 성분을 나타낸다.

초기조건 $u_{0,k} = v_{0,k} = 0.5$ 에 대해, 다음 과정을 3D 모델 근사 부분처럼 반복한다.

1. 식 (41)와 같이 \mathbf{b}_{11} 을 구한다.
2. 식 (42)와 같이 u_k, v_k 를 갱신한다.

이때, 베지어 곡면은 $0 \leq u, v \leq 1$ 의 범위에서 정의되므로 중간 과정에서 수렴성을 보장하기 위해 위와 비슷하게 $u_{n,k}$ 또는 $v_{n,k}$ 가 0.01보다 작아지면 0.01로 대체하고, 0.99보다 커지면 0.99로 대체한다.

위의 과정은 3D 모델 근사부분과 동일하다고 할 수 있다 하지만 위의 과정을 R함수 뿐만이 아닌 G,B함수에서도 해야하는 것과 오차함수 부분은 색상까지 표현하는 만큼 다르게 정의해야 한다.

Defintion II.12. 2D RGB저장 부분에서 근사시킨 R,G,B 함수를 각각 $\mathbf{Rx}_k, \mathbf{Gx}_k, \mathbf{Bx}_k$, 근사시키고 싶은 3D 모델에서 뽑은 점들의 R,G,B함수를 각각 $\mathbf{RP}_k, \mathbf{GP}_k, \mathbf{BP}_k$ 라고 할 때, R오차함수, G오차함수, B오차함수 μ_R, μ_G, μ_B 와 오차함수 μ 는 다음과 같이 주어진다.

$$\begin{aligned}\mu_R &= \max_{f_R \text{ 영역}} \max_{1 \leq k \leq K} \|\mathbf{Rx}_k - \mathbf{RP}_k\| \\ \mu_G &= \max_{f_G \text{ 영역}} \max_{1 \leq k \leq K} \|\mathbf{Gx}_k - \mathbf{GP}_k\| \\ \mu_B &= \max_{f_B \text{ 영역}} \max_{1 \leq k \leq K} \|\mathbf{Bx}_k - \mathbf{BP}_k\| \\ \mu &= \mu_R + \mu_G + \mu_B\end{aligned}\tag{43}$$

우리는 R 함수, G 함수,B 함수의 분할횟수 중 최솟값을 n° 이라고 하였을때 $n \rightarrow \infty$ 에 따라 $\mu \rightarrow 0$ 임을 보였다.

II.3.4 수렴성 증명

Theorem II.13. 위와 같은 근사 방법에 따라 f_R 을 근사했다고 하자. 오차함수 μ_R 에 대해 다음이 성립한다.

$$\lim_{n \rightarrow \infty} \mu_R = 0$$

Proof. $B_i^2(u)B_j^2(v)$ 의 합이 1이므로 다음이 성립한다.

$$\begin{aligned}\|\mathbf{Rx}_k - \mathbf{RP}_k\| &= \left\| \sum_{i,j=0}^2 B_i^2(u_k)B_j^2(v_k)(\mathbf{b}_{ij} - \mathbf{RP}_k) \right\| \\ &\leq \sum_{i,j=0}^2 B_i^2(u_k)B_j^2(v_k) \|\mathbf{b}_{ij} - \mathbf{RP}_k\|\end{aligned}$$

$\|\mathbf{x}_k - \mathbf{RP}_k\|$ 는 $\|\mathbf{b}_{ij} - \mathbf{RP}_k\|$ 의 가중평균이므로, $\|\mathbf{b}_{ij} - \mathbf{RP}_k\| \rightarrow 0$ 이면 $\|\mathbf{x}_k - \mathbf{RP}_k\| \rightarrow 0$ 이다.

여기서 f_R 은 2개의 독립변수 (x,y)에서 R의 색깔을 반환하는 함수이므로 $\mathbf{f}: [0,a] \times [0,b] \rightarrow [0,255]$ 라고 할 수 있고 이는 즉, 콤팩트 집합에서 정의된 연속함수이므로 Heine-Cantor Theorem에 의해 균등연속(uniformly continuous)이다. 임의의 $\varepsilon > 0$ 에 대해 다음을 만족하는 $\delta > 0$ 이 존재한다.

$$\sqrt{(x-x')^2 + (y-y')^2} < \delta \implies \|\mathbf{f}_R(x, y) - \mathbf{f}_R(x', y')\| < \varepsilon$$

○] 제 $\mathbf{RP}_k = \mathbf{f}_R(x_k, y_k)$ 라 하고, (x, y) 를 포함하는 영역

$$I_{ab} = \left[\frac{a}{2^n}, \frac{(a+1)}{2^n} \right] \times \left[\frac{b}{2^n}, \frac{b+1}{2^n} \right]$$

가 (x, y) 의 $\delta/2$ -근방에 포함되도록 하는 최소의 자연수 N 을 잡자. $\mathbf{b}_{00}, \mathbf{b}_{02}, \mathbf{b}_{20}, \mathbf{b}_{22} \in \mathbf{f}(I_{ab})$

○] 므로, $n \geq N$ 이면 $\|\mathbf{b}_{00} - \mathbf{RP}_k\|, \|\mathbf{b}_{02} - \mathbf{RP}_k\|, \|\mathbf{b}_{20} - \mathbf{RP}_k\|, \|\mathbf{b}_{22} - \mathbf{RP}_k\|$ 는 모두 ε 보다 작다. 또한 $\mathbf{f}(I_{ab})$ 가 \mathbf{b}_{00} 의 ε -근방에 포함되므로 $\|\mathbf{b}_{01} - \mathbf{RP}_k\| < \|\mathbf{b}_{01} - \mathbf{Q}\| + \|\mathbf{Q} - \mathbf{RP}_k\| < 2\varepsilon$ ○] 성립한다. 마찬가지로 $\|\mathbf{b}_{10} - \mathbf{RP}_k\|, \|\mathbf{b}_{12} - \mathbf{RP}_k\|, \|\mathbf{b}_{21} - \mathbf{RP}_k\|$ 는 모두 2ε 보다 작다. 따라서 $(i, j) \neq (1, 1)$ 에 대해 $\lim_{n \rightarrow \infty} \|\mathbf{b}_{ij} - \mathbf{RP}_k\| = 0$ ○] 다.

\mathbf{b}_{11} 은 (41) 과 같이 주어지고, ○] 때 \mathbf{y}_k 는 다음과 같다.

$$\mathbf{y}_k = \sum_{(i,j) \neq (1,1)} B_i^2(u_k) B_j^2(v_k) \mathbf{b}_{ij}$$

$\|\mathbf{b}_{11} - \mathbf{RP}_k\|$ 는 아래와 같은 부등식으로 계산된다.

$$\begin{aligned} & \|\mathbf{b}_{11} - \mathbf{RP}_k\| \\ &= \left\| \frac{\sum_{k=1}^K B_1^2(u_k) B_1^2(v_k) \sum_{(i,j) \neq (1,1)} B_i^2(u_k) B_j^2(v_k) (\mathbf{RP}_k - \mathbf{b}_{ij})}{\sum_{k=1}^K (B_1^2(u_k) B_1^2(v_k))^2} \right\| \\ &\leq \frac{\sum_{k=1}^K B_1^2(u_k) B_1^2(v_k) \sum_{(i,j) \neq (1,1)} B_i^2(u_k) B_j^2(v_k) \|\mathbf{RP}_k - \mathbf{b}_{ij}\|}{\sum_{k=1}^K (B_1^2(u_k) B_1^2(v_k))^2} \\ &\leq \frac{\sum_{k=1}^K B_1^2(u_k) B_1^2(v_k) (1 - B_1^2(u_k) B_1^2(v_k))}{\sum_{k=1}^K (B_1^2(u_k) B_1^2(v_k))^2} \cdot 2\varepsilon \\ &= \left[\frac{\sum_{k=1}^K B_1^2(u_k) B_1^2(v_k)}{\sum_{k=1}^K (B_1^2(u_k) B_1^2(v_k))^2} - 1 \right] \cdot 2\varepsilon \\ &\leq \frac{K}{\sum_{k=1}^K (B_1^2(u_k) B_1^2(v_k))^2} \frac{\varepsilon}{2} \end{aligned}$$

마지막 식에서 다음과 같은 절대부등식을 사용했다.

$$\frac{\sum_{k=1}^K x_k}{\sum_{k=1}^K x_k^2} \leq 1 + \frac{K}{4 \sum_{k=1}^K x_k^2}$$

이는 절대부등식 $x_k \leq x_k^2 + 1/4$ 를 변변이 더하고 $\sum x_k^2$ 으로 나누면 얻을 수 있다.

u_k 와 v_k 가 항상 $[0.01, 0.99]$ 범위에 있으므로 $(B_1^2(u_k)B_1^2(v_k))^2 \geq (2 \times 0.01 \times 0.99)^4 = C^\circ$ 고, 이에 따라 $\|\mathbf{b}_{11} - \mathbf{R}\mathbf{P}_k\| \leq \varepsilon K/2C^\circ$ 이다. 여기서 K 는 영역 안에 있는 점의 개수이다. 전체 점 개수를 K_1 라 한다면 $\|\mathbf{b}_{11} - \mathbf{R}\mathbf{P}_k\| \leq \varepsilon K/2C \leq \varepsilon K_1/2C^\circ$ 임을 알 수 있다. 모든 $0 \leq i, j \leq 2$ 에 대해 $\lim_{n \rightarrow \infty} \|\mathbf{b}_{ij} - \mathbf{R}\mathbf{P}_k\| = 0^\circ$ 므로, 이들의 가중평균인 $\mathbf{x}_k - \mathbf{R}\mathbf{P}_k$ 도 $n \rightarrow \infty$ 의 극한에서 0으로 수렴한다.

각 k 에 대해서 $\lim_{n \rightarrow \infty} \|\mathbf{x}_k - \mathbf{R}\mathbf{P}_k\| = 0^\circ$ 므로, 모든 $\varepsilon > 0$ 에 대해 적당한 자연수 N_k 가 존재해 $n \geq N_k \implies \|\mathbf{x}_k - \mathbf{R}\mathbf{P}_k\| < \varepsilon_R^\circ$ 을 만족한다. 이제 자연수 N_R 을 N_k 들의 최댓값으로 정의한다.

$$N_R = \max_{\text{각 영역 } 1 \leq k \leq K} N_k$$

그러면 모든 $n \geq N_R$ 에 대해 $\mu < \varepsilon_R^\circ$ 므로, $\lim_{n \rightarrow \infty} \mu_R = 0$ 을 얻는다. \square

우리는 정리 II.13에 따라 다음을 얻을 수 있다.

Theorem II.14. 우리가 R 함수, G 함수, B 함수를 위와 같이 근사했다고 가정하자. 그러면 오차함수 μ 에 대해서 다음이 성립한다.

$$\lim_{n \rightarrow \infty} \mu = 0$$

Proof. 우리는 오차함수 μ 를 다음과 같이 정의했었다.

$$\mu = \mu_R + \mu_G + \mu_B$$

근데 우리는 μ_R° 이 0으로 수렴함을 증명하였다. 비슷하게 μ_G, μ_B 도 0으로 수렴함을 알 수 있으므로 μ 도 수렴함을 알 수 있다. \square

이 과정을 통해 우리는 분할 횟수를 늘릴 때 항상 오차함수가 0으로 수렴함을 보일 수 있었다.

II.4 3D model RGB implementation

2차원에서 색상이 있을 때 근사하는 방법을 알아보았다면 이번에는 3차원에서 색상이 있을 때 근사하는 방법을 알아볼 것이다. 2차원일 때는 이에 해당되는 R,G,B 값을 하나의 축으로 하여 3차원 그래프 상에서 근사하였다면 이번에는 약간 다르게 새로운 벡터를 정의하고 그 것을 3차원 모델 근사처럼 비슷하게 가져갈 것이다.

Defintion II.15. 만약 점 \mathbf{P} 에 색상 벡터 \mathbf{Q} 가 대응된다면 우리는 \mathbf{P}, \mathbf{Q} 순서대로 열에 배열한 벡터 \mathbf{K} 를 색상 위치 벡터라고 정의한다.

즉 우리는 색상 위치 벡터들의 모임을 가중치 베지어 곡면을 이용하여 근사할 것이다. 위의 근사 방법과 약간 다른 점은 위치 벡터가 색상 위치 벡터인 것이고 베지어 곡면으로 근사하는 대신 가중치 베지어 곡면으로 근사한다는 것이다. 이러한 점에서 분할하는 방법은 가장 거리가 먼 2개의 점을 z축에 배치하고 원통형 분할을 하던 것 그대로 가져갈 것이다. 하지만 가중치 베지어 곡면의 RGB 조절점에 대한 것은 약간 다른 점이 있어 조절점을 정하는 방법을 알아볼 것이다.

II.4.1 Control Point Determination

이제 원통형 분할로 나뉘어진 영역에 존재하는 색상 위치 벡터를 (2,2)차 가중치 베지어 곡면으로 근사해야 한다. 가중치 베지어 곡면을 결정하기 위해서는 RGB 조절점 9개를 잡아야 한다. 일단 가중치 베지어 곡면이 연속적으로 이어지기 위해서는 \mathbf{k}_{11} 를 제외한 나머지 조절점은 영역 안에 있는 정점들과는 무관하게 결정되어야 한다. 각 영역의 경계는 $\theta = \theta_i$ 혹은 $z = z_j$ 로 주어지므로, 직선 $\ell: \theta = \theta_i, z = z_j$ 와 obj 파일의 면 F 의 교점을 네 조절점 $\mathbf{k}_{00}, \mathbf{k}_{02}, \mathbf{k}_{20}, \mathbf{k}_{22}$ 를 구한다. 여기서는 직선과 삼각형의 교점을 찾는 빠른 알고리즘인 Möller-Trumbore intersection algorithm을 사용한다. 그리고 여기서 이 알고리즘은 교점을 얻을 수 있는 알고리즘이지만 이 알고리즘에 있는 위치 벡터를 색상 위치 벡터로 바꿔준다면 교점이

색상 위치 벡터 꼴로 나와 우리는 이 합계점을 극복할 수 있다. [8] 우리는 2D model 근사에서 중간 조절점을 잡은 방법을 이용하여 네 조절점 $\mathbf{k}_{01}, \mathbf{k}_{10}, \mathbf{k}_{12}, \mathbf{k}_{21}$ 을 얻을 수 있다. 여기서 색상벡터도 결국 2차 베지어 곡선을 따르므로 중간 조절점을 얻는 방법을 가중치 베지어 곡선에 적용하여 색상조절점 벡터를 이용하여 구해도 무관하다. 이에 대한 그림은 앞에서와 같이 Fig. 12부분에 있으므로 생략한다.

\mathbf{k}_{11} 을 구하기 위해 최소제곱법을 이용한다. 최소제곱법을 사용하기 위해서는 obj.mtl 파일의 각 색상 위치 벡터 \mathbf{P}_k 에 대응되는 가중치 베지어 곡면 위의 색상위치벡터 $\mathbf{x}_k = \mathbf{x}(u_k, v_k)$ 을 알아야 한다. 이미 알고 있는 8개 RGB 조절점에 대한 항을 \mathbf{y}_k 라 하면 다음과 같이 나타낼 수 있다.

$$\mathbf{x}_k = B_1^2(u_k)B_1^2(v_k)\mathbf{k}_{11} + \mathbf{y}_k$$

이제 $\mathbf{x}_k = \mathbf{P}_k$ ($k = 1, \dots, K$)의 제곱오차 $S = \sum_{k=1}^K \|\mathbf{x}_k - \mathbf{P}_k\|^2$ 를 최소화하기 위해 행렬방정식으로 나타낸다.

$$\begin{aligned} & \begin{pmatrix} B_1^2(u_1)B_1^2(v_1) \\ B_1^2(u_2)B_1^2(v_2) \\ \vdots \\ B_1^2(u_K)B_1^2(v_K) \end{pmatrix} \begin{pmatrix} k_{11}^x & k_{11}^y & k_{11}^z & k_{11}^R & k_{11}^G & k_{11}^B \end{pmatrix} \\ &= \begin{pmatrix} P_1^x - y_1^x & P_1^y - y_1^y & P_1^z - y_1^z & P_1^R - y_1^R & P_1^G - y_1^G & P_1^B - y_1^B \\ P_2^x - y_2^x & P_2^y - y_2^y & P_2^z - y_2^z & P_2^R - y_2^R & P_2^G - y_2^G & P_2^B - y_2^B \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ P_K^x - y_K^x & P_K^y - y_K^y & P_K^z - y_K^z & P_K^R - y_K^R & P_K^G - y_K^G & P_K^B - y_K^B \end{pmatrix} \end{aligned}$$

이때 윗첨자 x, y, z 는 각각 벡터의 x, y, z 성분을 가르키고 윗첨자 R, G, B 는 각각 색상 벡터의 R, G, B 성분을 가르킨다. 이 경우에도 제곱오차는 마찬가지로 S 이다. $AX = B$ 꼴의 행렬방정식에서 최소제곱오차를 가지는 \hat{X} 는 $\hat{X} = (A^\top A)^{-1}A^\top B$ 로 주어지므로, \mathbf{k}_{11} 을 다음과 같이 근사할 수 있다.

$$\mathbf{k}_{11} = \frac{\sum_{k=1}^K B_1^2(u_k)B_1^2(v_k)(\mathbf{P}_k - \mathbf{y}_k)}{\sum_{k=1}^K (B_1^2(u_k)B_1^2(v_k))^2} \quad (44)$$

각 k 에 대해 u_k 와 v_k 의 값을 안다면 Eq. (44)과 같이 \mathbf{k}_{11} 을 구할 수 있다. 이제 최적의 u_k, v_k 를

찾기 위해 뉴턴-랩슨 방법을 이용한다. 뉴턴-랩슨 방법을 적용할 함수는 $\mathbf{x} - \mathbf{P}_k$ 이다. 즉 현재 $u_{n,k}$ 와 $v_{n,k}$ 가 주어질 때, 다음과 같이 $u_{n+1,k}$ 와 $v_{n+1,k}$ 를 얻는다.

$$u_{n+1,k} = u_{n,k} - \frac{\mathbf{x} - \mathbf{P}_k}{\partial(\mathbf{x} - \mathbf{P}_k)/\partial u} \Big|_{u=u_{n,k}} \quad (45)$$

$$v_{n+1,k} = v_{n,k} - \frac{\mathbf{x} - \mathbf{P}_k}{\partial(\mathbf{x} - \mathbf{P}_k)/\partial v} \Big|_{v=v_{n,k}} \quad (46)$$

그런데 벡터의 나눗셈은 정의되지 않으므로 이번에도 최소제곱법을 이용한다. 즉, $[\partial(\mathbf{x} - \mathbf{P}_k)/\partial u]\Delta u = \mathbf{x} - \mathbf{P}_k$ 의 제곱 오차를 최소로 하는 스칼라 Δu 를 선택한다.

초기 조건 $u_{0,k} = v_{0,k} = 0.5$ 에 대해 \mathbf{k}_{11} 를 구하고 새로 u_k, v_k 를 구하는 과정을 반복한다.

II.4.2 Algorithm

일단 어떠한 3D RGB 모델이 있을 때 위에서 말했던 것처럼 축을 돌리고 이를 원통형 분할을 할 것이다.

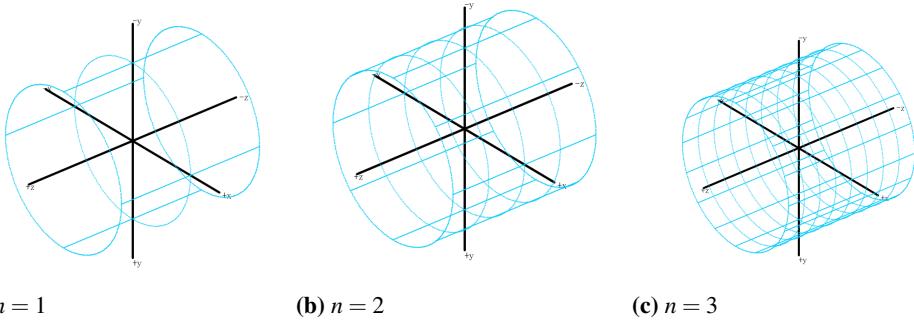


Figure 20. 원통형 분할

원통형 분할은 θ 와 z 의 범위를 이등분하여 이루어진다. 그림에서는 θ 와 z 의 범위를 각각 $2^{n+1}, 2^n$ 등분했다.

분할된 영역의 곡면을 하나의 가중치 베지어 곡면으로 근사한다. 각 영역의 가중치 베지어 곡면이 연속적으로 이어지기 위해서는 베지어 곡면의 네 모서리 ($u=0, u=1, v=0, v=1$) 가 영역의 경계에 위치해야 한다. 특히 네 RGB 조절점 $\mathbf{k}_{00} = \mathbf{x}(0,0), \mathbf{k}_{02} = \mathbf{x}(0,1), \mathbf{k}_{20} =$

$\mathbf{k}(1,0), \mathbf{k}_{22} = \mathbf{x}(1,1)$ 는 각각 4개의 영역이 만나는 경계에 위치한다.

분할된 영역이 $\theta_a \leq \theta \leq \theta_{a+1}, z_b \leq z \leq z_{b+1}$ 로 주어진다고 하자. 반직선 $\ell_{ab}: \theta = \theta_a, z = z_b$ 로 정의하고, 네 반직선 $\ell_{ab}, \ell_{a,b+1}, \ell_{a+1,b}, \ell_{a+1,b+1}$ 과 obj파일의 면 F 의 교점을 찾아 각각 $\mathbf{k}_{00}, \mathbf{k}_{02}, \mathbf{k}_{20}, \mathbf{k}_{22}$ 로 둔다. 이때 광선 또는 직선과 삼각형의 교점을 방법인 Möller-Trumbore intersection algorithm을 이용한다. [8]

Möller-Trumbore intersection algorithm을 이용하면 영역의 경계 $\theta_a \leq \theta \leq \theta_{a+1}, z = z_b$ 와 F 의 교선을 찾을 수 있다. 그리고 앞에서 말한 이유 때문에 가중치 베지어 곡선에도 다음 2차 베지어 곡선의 성질(II.4)을 이용할 수 있었다.

따라서 영역의 경계 $\theta_a \leq \theta \leq \theta_{a+1}, z = z_b$ 와 ∂C 의 교선을 잡고, 교선 위의 점들 중 \mathbf{k}_{00} 과 \mathbf{k}_{02} 를 잇는 직선에서 가장 멀리 떨어진 점에 대응되는 색상 위치 벡터 \mathbf{Q} 를 찾는다. 그러면 $\mathbf{k}_{01} = 2\mathbf{Q} - (\mathbf{k}_{00} + \mathbf{k}_{02})/2$ 로 주어진다. 같은 방법으로 조절점 $\mathbf{k}_{01}, \mathbf{k}_{10}, \mathbf{k}_{12}, \mathbf{k}_{21}$ 을 얻을 수 있다.

이제 RGB 조절점 \mathbf{k}_{11} 을 구하기 위해 최소제곱법과 가우스-뉴턴 방법을 이용한다. 이 아이디어는 선형 연구를 참고했다. [10] 영역 내의 색상 위치 벡터를 \mathbf{P}_k ($k = 1, 2, \dots, K$)라 하자. 각 \mathbf{P}_k 에 대응되는 가중치 베지어 곡면의 매개변수 u_k 와 v_k 가 주어졌다고 가정한다. $\mathbf{x}_k = \mathbf{x}(u_k, v_k)$ 에서 이미 알고 있는 RGB 조절점들에 대한 항을 \mathbf{y}_k 로 두면 다음과 같이 나타날 수 있다.

$$\mathbf{x}_k = \mathbf{x}(u_k, v_k) = B_1^2(u_k)B_1^2(v_k)\mathbf{k}_{11} + \mathbf{y}_k$$

\mathbf{k}_{11} 에 관한 연립방정식 $\mathbf{x}_k = \mathbf{P}_k$ ($1 \leq k \leq K$)는 일반적으로 해를 가지지 않는다. 따라서 제곱오차 $S = \sum_{k=1}^K \|\mathbf{x}_k - \mathbf{P}_k\|^2$ 을 최소로 만드는 \mathbf{k}_{11} 의 값을 구한다. 이를 위해 x, y, z, R, G, B

좌표로 나눠 행렬방정식으로 표현한다.

$$\begin{aligned}
 & \begin{pmatrix} B_1^2(u_1)B_1^2(v_1) \\ B_1^2(u_2)B_1^2(v_2) \\ \vdots \\ B_1^2(u_K)B_1^2(v_K) \end{pmatrix} (k_{11}^x \quad k_{11}^y \quad k_{11}^z \quad k_{11}^R \quad k_{11}^G \quad k_{11}^B) \\
 &= \begin{pmatrix} P_1^x - y_1^x & P_1^y - y_1^y & P_1^z - y_1^z & P_1^R - y_1^R & P_1^G - y_1^G & P_1^B - y_1^B \\ P_2^x - y_2^x & P_2^y - y_2^y & P_2^z - y_2^z & P_2^R - y_2^R & P_2^G - y_2^G & P_2^B - y_2^B \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ P_K^x - y_K^x & P_K^y - y_K^y & P_K^z - y_K^z & P_K^R - y_K^R & P_K^G - y_K^G & P_K^B - y_K^B \end{pmatrix}
 \end{aligned}$$

이때 윗첨자 x, y, z 는 각각 벡터의 x, y, z 성분, R, G, B 는 각각 색상 벡터의 R, G, B 성분을 나타낸다. 위 행렬방정식의 제곱오차도 마찬가지로 S 임을 알 수 있다. $AX = B$ 꼴의 행렬방정식에서 최소제곱오차를 가지는 $X = (A^\top A)^{-1}A^\top B$ 로 주어지므로, 다음과 같이 \mathbf{k}_{11} 을 구할 수 있다.

$$\mathbf{k}_{11} = \frac{\sum_{k=1}^K B_1^2(u_k)B_1^2(v_k)(\mathbf{P}_k - \mathbf{y}_k)}{\sum_{k=1}^K (B_1^2(u_k)B_1^2(v_k))^2} \quad (47)$$

각 \mathbf{P}_k 에 대해 대응되는 u_k 와 v_k 를 알고 있다면 (47)과 같이 \mathbf{k}_{11} 을 구할 수 있다. 이제 최적의 u_k 와 v_k 를 얻기 위해 가우스-뉴턴 방법을 사용한다. 즉, 각각의 k 에 대해 \mathbf{x}_k 와 \mathbf{P}_k 의 차이를 줄이는 방향으로 u_k 와 v_k 를 갱신한다. 현재 $u_{n,k}$ 와 $v_{n,k}$ 의 값이 $\mathbf{u}_{n,k} = (u_{n,k}, v_{n,k})^\top$ 로 주어질 때, 다음과 같이 $\mathbf{u}_{n+1,k}$ 를 알 수 있다.

$$\mathbf{u}_{n+1,k} = \mathbf{u}_{n,k} - (\mathbf{J}_x^\top \mathbf{J}_x)^{-1} \mathbf{J}_x^\top (\mathbf{x}_k - \mathbf{P}_k) \quad (48)$$

이때 \mathbf{J}_x 는 $\mathbf{u} = (u, v)^\top$ 에 대한 \mathbf{x} 의 Jacobian matrix이다.

$$\mathbf{J}_x = \begin{pmatrix} \partial x^x / \partial u & \partial x^x / \partial v \\ \partial x^y / \partial u & \partial x^y / \partial v \\ \partial x^z / \partial u & \partial x^z / \partial v \\ \partial x^R / \partial u & \partial x^R / \partial v \\ \partial x^G / \partial u & \partial x^G / \partial v \\ \partial x^B / \partial u & \partial x^B / \partial v \end{pmatrix}$$

앞에서와 마찬가지로 윗첨자는 벡터의 성분을 나타낸다.

초기조건 $u_{0,k} = v_{0,k} = 0.5$ 에 대해, \mathbf{k}_{11} 를 구하고 u_k, v_k 를 갱신하는 과정을 반복한다.

이때, 베지어 곡면은 $0 \leq u, v \leq 1$ 의 범위에서 정의되므로 중간 과정에서 $u_{n,k}$ 또는 $v_{n,k}$ 가 0.01보다 작아지면 0.01로 대체하고, 0.99보다 커지면 0.99로 대체한다. 가우스-뉴턴 방법은 수렴성을 보장할 수 없기 때문에, 이러한 보정은 $u_{n,k}$ 또는 $v_{n,k}$ 가 발산하지 않도록 한다.

그러면 이제 가중치 베지어 곡면으로 근사하는 과정을 알아보았으니 새로운 오차함수에 대해 알아보아야 한다.

Defintion II.16. RGB 저장 부분에서 오차함수는 다음과 같이 정의한다. 분할된 각 영역 내의 색상 위치 벡터를 \mathbf{P}_k ($1 \leq k \leq K$)라고 하고, 이에 대응되는 가중치 베지어 곡면 위의 색상 위치 벡터를 $\mathbf{x}(u_k, v_k) = \mathbf{x}_k$ 라 하자. 이때 오차함수 μ 는 다음과 같이 주어진다.

$$\mu = \max_{\text{각 영역}} \max_{1 \leq k \leq K} \|\mathbf{x}_k - \mathbf{P}_k\| \quad (49)$$

우리는 $n \rightarrow \infty$ 에 따라 $\mu \rightarrow 0$ 임을 보였다.

II.4.3 수렴성 증명

Theorem II.17. 위와 같은 근사 방법에 따라 3D 모델을 근사했다고 하자. 식 (49)로 주어진 오차함수 μ 에 대해 다음이 성립한다.

$$\lim_{n \rightarrow \infty} \mu = 0$$

Proof. $B_i^2(u)B_j^2(v)$ 의 합이 1이므로 다음이 성립한다.

$$\begin{aligned} \|\mathbf{x}_k - \mathbf{P}_k\| &= \left\| \sum_{i,j=0}^2 B_i^2(u_k)B_j^2(v_k)(\mathbf{k}_{ij} - \mathbf{P}_k) \right\| \\ &\leq \sum_{i,j=0}^2 B_i^2(u_k)B_j^2(v_k) \|\mathbf{k}_{ij} - \mathbf{P}_k\| \end{aligned}$$

$\|\mathbf{x}_k - \mathbf{P}_k\|$ 은 $\|\mathbf{k}_{ij} - \mathbf{P}_k\|$ 의 가중평균이므로, $\|\mathbf{k}_{ij} - \mathbf{P}_k\| \rightarrow 0$ 이면 $\|\mathbf{x}_k - \mathbf{P}_k\| \rightarrow 0$ 이다.

정리 II.8의 함수 \mathbf{f} 를 생각하자. $\mathbf{f}: [0, 2\pi] \times [-h, h] \rightarrow \mathbb{R}^3$ 는 콤팩트 집합에서 정의된 연속 함수이므로 Heine-Cantor Theorem에 의해 균등연속(uniformly continuous)이다. 즉, 임의의 $\varepsilon > 0$ 에 대해 다음을 만족하는 $\delta > 0$ 이 존재한다.

$$\sqrt{(\theta - \theta')^2 + (z - z')^2} < \delta \implies \|\mathbf{f}(\theta, z) - \mathbf{f}(\theta', z')\| < \varepsilon$$

○] 제 $\mathbf{P}_k = \mathbf{f}(\theta_k, z_k)$ 라 하고, (θ_k, z_k) 를 포함하는 영역

$$I_{ab} = \left[\frac{2\pi a}{2^n}, \frac{2\pi(a+1)}{2^n} \right] \times \left[-h + \frac{2hb}{2^n}, -h + \frac{2h(b+1)}{2^n} \right]$$

가 (θ_k, z_k) 의 $\delta/2$ -근방에 포함되도록 하는 최소의 자연수 N 을 잡자. $\mathbf{k}_{00}, \mathbf{k}_{02}, \mathbf{k}_{20}, \mathbf{k}_{22} \in \mathbf{f}(I_{ab})$ 이므로, $n \geq N$ 이면 $\|\mathbf{k}_{00} - \mathbf{P}_k\|, \|\mathbf{k}_{02} - \mathbf{P}_k\|, \|\mathbf{k}_{20} - \mathbf{P}_k\|, \|\mathbf{k}_{22} - \mathbf{P}_k\|$ 는 모두 ε 보다 작다. 또 한 $\mathbf{f}(I_{ab})$ 가 \mathbf{k}_{00} 의 ε -근방에 포함되므로 $\|\mathbf{k}_{01} - \mathbf{P}_k\| < \|\mathbf{k}_{01} - \mathbf{Q}\| + \|\mathbf{Q} - \mathbf{P}_k\| < 2\varepsilon$ 성립한다. 마찬가지로 $\|\mathbf{k}_{10} - \mathbf{P}_k\|, \|\mathbf{k}_{12} - \mathbf{P}_k\|, \|\mathbf{k}_{21} - \mathbf{P}_k\|$ 는 모두 2ε 보다 작다. 따라서 $(i, j) \neq (1, 1)$ 에 대해 $\lim_{n \rightarrow \infty} \|\mathbf{k}_{ij} - \mathbf{P}_k\| = 0$ 이다.

\mathbf{k}_{11} 은 (47)과 같이 주어지고, 이 때 \mathbf{y}_k 는 다음과 같다.

$$\mathbf{y}_k = \sum_{(i,j) \neq (1,1)} B_i^2(u_k) B_j^2(v_k) \mathbf{k}_{ij}$$

$\|\mathbf{k}_{11} - \mathbf{P}_k\|$ 는 아래와 같은 부등식으로 계산된다.

$$\begin{aligned} & \|\mathbf{k}_{11} - \mathbf{P}_k\| \\ &= \left\| \frac{\sum_{k=1}^K B_1^2(u_k) B_1^2(v_k) \sum_{(i,j) \neq (1,1)} B_i^2(u_k) B_j^2(v_k) (\mathbf{P}_k - \mathbf{k}_{ij})}{\sum_{k=1}^K (B_1^2(u_k) B_1^2(v_k))^2} \right\| \\ &\leq \frac{\sum_{k=1}^K B_1^2(u_k) B_1^2(v_k) \sum_{(i,j) \neq (1,1)} B_i^2(u_k) B_j^2(v_k) \|\mathbf{P}_k - \mathbf{k}_{ij}\|}{\sum_{k=1}^K (B_1^2(u_k) B_1^2(v_k))^2} \\ &\leq \frac{\sum_{k=1}^K B_1^2(u_k) B_1^2(v_k) (1 - B_1^2(u_k) B_1^2(v_k))}{\sum_{k=1}^K (B_1^2(u_k) B_1^2(v_k))^2} \cdot 2\varepsilon \\ &= \left[\frac{\sum_{k=1}^K B_1^2(u_k) B_1^2(v_k)}{\sum_{k=1}^K (B_1^2(u_k) B_1^2(v_k))^2} - 1 \right] \cdot 2\varepsilon \\ &\leq \frac{K}{\sum_{k=1}^K (B_1^2(u_k) B_1^2(v_k))^2} \frac{\varepsilon}{2} \end{aligned}$$

마지막 식에서 다음과 같은 절대부등식을 사용했다.

$$\frac{\sum_{k=1}^K x_k}{\sum_{k=1}^K x_k^2} \leq 1 + \frac{K}{4 \sum_{k=1}^K x_k^2}$$

이는 절대부등식 $x_k \leq x_k^2 + 1/4$ 를 변변이 더하고 $\sum x_k^2$ 으로 나누면 얻을 수 있다.

u_k 와 v_k 가 항상 $[0.01, 0.99]$ 범위에 있으므로 $(B_1^2(u_k)B_1^2(v_k))^2 \geq (2 \times 0.01 \times 0.99)^4 = C$ 이고, 이에 따라 $\|\mathbf{k}_{11} - \mathbf{P}_k\| \leq \varepsilon K / 2C$ 이다. 여기서 K 는 영역 안에 있는 점의 개수지만 전체 점 개수를 K_1 라 한다면 $\|\mathbf{k}_{11} - \mathbf{P}_k\| \leq \varepsilon K / 2C \leq \varepsilon K_1 / 2C$ 때문에 굳이 신경 쓸 필요 없다. 모든 $0 \leq i, j \leq 2$ 에 대해 $\lim_{n \rightarrow \infty} \|\mathbf{k}_{ij} - \mathbf{P}_k\| = 0$ 므로, 이들의 가중평균인 $\mathbf{x}_k - \mathbf{P}_k$ 도 $n \rightarrow \infty$ 의 극한에서 0으로 수렴한다.

각 k 에 대해서 $\lim_{n \rightarrow \infty} \|\mathbf{x}_k - \mathbf{P}_k\| = 0$ 으로, 모든 $\varepsilon > 0$ 에 대해 적당한 자연수 N_k 가 존재해 $n \geq N_k \implies \|\mathbf{x}_k - \mathbf{P}_k\| < \varepsilon$ 을 만족한다. 이제 자연수 N 을 N_k 들의 최댓값으로 정의한다.

$$N = \max_{\text{각 영역}} \max_{1 \leq k \leq K} N_k$$

그러면 모든 $n \geq N$ 에 대해 $\mu < \varepsilon$ 므로, $\lim_{n \rightarrow \infty} \mu = 0$ 을 얻는다. \square

이 과정을 통해 위치 벡터를 색상 위치 벡터로 확장시킴으로서 비슷하게 3차원 모델근사방법을 사용할 수 있었고 이런 근사 과정을 통해 3차원 RGB 모델도 근사 가능함을 보일 수 있었다.

II.5 Model Movement Implementation

위의 과정을 통해 항상 조절점을 아는 베지어 곡선 베지어 곡면의 움직임을 구현할 수 있다. 베지어 곡선과 베지어 곡면을 움직인다는 것은 상을 바꾼다는 것이고 이는 조절점에 따라 달라지므로 조절점이 이동하면 충분히 가능한 일이다. 만약 곡선이라면 시작 베지어 곡선과 끝 베지어 곡선, 베지어 곡면이라면 시작 베지어 곡면과 끝 베지어 곡면의 조절점을 안다면 조절점을 시작하는 곡선 곡면것에서 끝나는 곡선, 곡면 것으로 대응 시킨다면 된다. 만약 그

조절점이 \mathbf{b}_0 에서 \mathbf{b}_1 으로 대응된다면 시점 λ 에서 $\mathbf{b}_\lambda = (1 - \lambda)\mathbf{b}_0 + \lambda\mathbf{b}_1$ 을 정의한다면 \mathbf{b}_λ 들을 통해 그려진 베지어 곡선, 곡면은 시점 λ 에서의 중간 곡선 곡면이 된다. 가중치 베지어 곡선, 가중치 베지어 곡면이라도 위치 벡터 대신 색상 위치 벡터를 대입하면 충분히 움직임을 나타낼 수 있다.

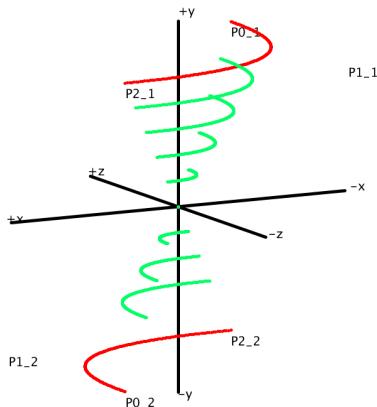


Figure 21. 베지어 곡선의 움직임

처음 곡선과 끝 곡선(빨간색)에 대해 이들을 연속적으로 변화시켜 중간 곡선(초록색)을 만든다.

다만 이 작업에서 자연스럽게 이동하게 만들어주지만 너무 앞과 끝에 집중되어 맞춰져 있다보니 중간에 원하는 부분이 안 나올 수도 있다. 그것은 움직임이 별로 크지 않도록 프레임을 나눠 사이사이를 다음과 같은 방법을 적용하는 식으로 사용하여 문제를 해결 할 수 있다.

III. Code Implementation

III.1 Lambertian Reflectance

프로세싱에서는 명암이 있지 않는다면 단색으로 덮어지는 경향이 있어 3차원 좌표계의 위치를 잘 파악하기 힘들다. 특정한 경우에는 명암을 넣지 않았지만 구의 obj 파일 같이 특정한 파일은 단색에 점들도 아주 많아서 위치를 구별하기 어려웠고 이를 해결하는 것은 필수적이었다. 그래서 많이 사용되는 조명 기법인 Lambertian reflectance를 사용하였다. 이는 실제로

mtl 파일에서 사용되는 기법이다.

빛의 효과를 받은 물체의 색깔은 전체적으로 3가지 모습의 합쳐진 모습으로 나타난다.

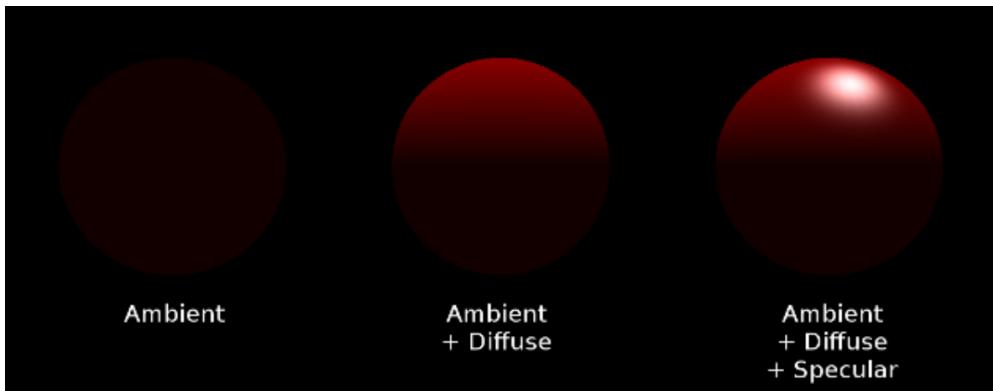


Figure 22. Lambertian reflectance

ambient color, diffuse color, specular color의 효과를 보여준다. 여기서 ambient color는 평균적인 색상을 의미하는데 전체적으로 어두워 잘 보이지 않지만 잘보면 평균적인 색상을 나타낸다는 것을 알 수 있다.

- 전체적인 평균 색상을 알려주는 ambient 색깔
- 물체의 조명의 위치에 따라 부분마다 다른 색깔이 나타날 것이다. 이에 대한 효과는 diffuse 색깔로 나타난다.
- 그리고 마지막으로 그 물체의 특수한 재질때문에 생기는 효과로 인한 색깔이 더해진다. 제일 대표적인 예시로 광택이 있으며 이를 specular 색깔이라고 한다.

각 ambient 색깔, diffuse 색깔, specular 색깔은 각각 K_a, K_d, K_s 라 하는 ambient 반사도, diffuse 반사도, specular 반사도에 영향을 받는다.

이 프로세싱에서 3D 좌표계의 가장 큰 문제는 광택처럼 재질을 표현할수 없는 것이 아니라 부분마다 똑같은 색이 아닌 다른 색이 나타나게 하는 것이다. 그래서 다른 부분은 너무 어려워 다루지 않았고 ambient 반사도, diffuse 반사도에 해당되는 내용만 다루었다.

이에 대한 식은 다음과 같다.

1. ambient 색깔만 반영하면 최종 색깔은 다음과 같은 식으로 나타난다.

$$\text{ambientcolor} = \text{materialcolor} * \text{ambient},$$

2. diffuse 색깔까지 반영하면 최종 색깔은 다음과 같은 식으로 나타난다.

$$\text{lightvector} = \text{lightposition} - \text{objectposition}$$

$$\text{cosine} = \text{dot product}(\text{objectnormalvector}(0), \text{normalizedlightvector})$$

$$\text{lambertfactor} = \max(\text{cosine}, 0)$$

$$\text{luminosity} = \frac{1}{1 + \text{distance} * \text{distance}}$$

$$\text{diffusecolor} = \text{materialcolor} * \text{lightcolor} * \text{lambertfactor} * \text{luminosity}$$

$$\text{finalcolor} = \text{ambientcolor} + \text{diffusecolor}$$

이에 대한 부분은 아래 Section III.3 부분에서 확인할 수 있을 것이다.

III.2 Arc Approximation Program

이 코드 구현 부분은 2D model improvement 의 알고리즘 부분 Section II.1.3의 원호 근사 부분 코드이다. 코드 내 함수는 다음과 같다.

```
#arc_approximation functions  
coordinate3D()  
point3D(float x,float y,float z)  
point3D_RGB(float x,float y,float z,float R,float G,float B,float st)  
line3D_RGB(float x,float y,float z,float w,float m,float n,float R,float G,  
float B)  
check3D(float x,float y,float z)  
Bezier2D(float x0,float y0,float x1,float y1,float x2,float y2,float num)
```

```

Bezier3D(float x0,float y0,float z0,float x1,float y1,float z1,float x2,
float y2,float z2, float num)

Bezier_line3D(float x0,float y0,float z0,float x1,float y1,float z1
, float x2, float y2, float z2, float x3, float y3, float z3, float x4, float y4,
float z4, float x5, float y5, float z5)

Bezier_move2D(float x0,float y0,float x1,float y1,float x2,float y2,float x3,
float y3, float x4, float y4, float x5, float y5)

Bezier_move3D(float x0,float y0,float z0,float x1,float y1,float z1,float x2,
float y2, float z2, float x3, float y3, float z3, float x4, float y4, float z4,
float x5, float y5, float z5)

Bezier_move_change3D(float x0,float y0,float z0,float x1,float y1,float z1,
float x2, float y2, float z2, float x3, float y3, float z3, float x4, float y4,
float z4, float x5, float y5, float z5, float k, float R, float G, float B, float m)

circle2D(float r, float n, float start, float last)

makeBezier(float start, float last, float r, float n)

goBezier(float start, float last, float n, float r, float num)

nogoBezier(float start, float last, float n, float r, float num)

noBezier(float x0, float y0, float x1, float y1, float x2, float y2)

Housedorf(float r, float start, float last, float x0, float y0, float x1, float y1,
float x2, float y2, float num)

```

이것을 프로세싱에 실행시킨뒤 0을 누르면 베지어 곡선이, 1을 누르면 움직이는 베지어 곡선이, 2를 누르면 근사된 결과가 나온다. 근사시킨 것은 2번째줄에서 app라는 근사 기준치를 기준으로 한다. 즉, 여기서 20이라는 것은 오차함수가 20 이하가 되면 멈추게 되는 것이다.

0,1을 누른상태에서 창을 우클릭하고 움직이면 좌표계가 회전하는 것을 볼 수 있다.

```

breakatwhitespace
int oneloop=0;
float t,move=0,app=20,maxmin,min,ceta,pi,ex=0;
float [] M = new float [100];
float [] H = new float [1000] [1000];
float [] A = new float [6];
void setup(){ size(1000,1000);background(255);}
void draw(){
  if(key=='0'){ex=0;oneloop=0;}if(key=='1'){ex=1;oneloop=0;}if(key=='2'){ex=2;oneloop=0;}
  if(ex==0){if(oneloop==0){background(255);oneloop=1;}}
  if(mouseButton==LEFT){
    background(0);
    ceta=mouseX/1000.0*360.0;pi=mouseY/1000.0*360.0;
    strokeWeight(5);stroke(255);
    textSize(20);text("pi=",85,100);text("ceta=",85,130);
    text(pi,120,100);text(ceta,140,130);
    coordinate3D();
    Bezier3D(100,120,250,250,200,30,10,120,30,1);
    Bezier3D(-100,-120,-250,-250,-200,-30,-10,-120,-30,1);
    Bezier_line3D
    (100,120,250,250,200,30,10,120,30,-100,-120,-250,-250,-200,-30,-10,-120,-30,1);
    Bezier_move3D
    (100,120,250,250,200,30,10,120,30,-100,-120,-250,-250,-200,-30,-10,-120,-30,1);
    if(ex==1){if(oneloop==0){background(255);oneloop=1;}}
    if(mouseButton==LEFT){
      background(0);
      ceta=mouseX/1000.0*360.0;pi=mouseY/1000.0*360.0;
      strokeWeight(5);stroke(255);
      textSize(20);text("pi=",85,100);text("ceta=",85,130);
      text(pi,120,100);text(ceta,140,130);
      coordinate3D();
      Bezier_move_change3D
      (100,120,250,250,200,30,10,120,30,-100,-120,-250,-250,-200,-30,-10,-120,-30,1);
      Bezier_move_change3D
      (100,120,250,250,200,30,10,120,30,-100,-120,-250,-250,-200,-30,-10,-120,-30,1);
      move=0.25*move,255-255*move,0,10);
      if(move<0.0,0.0,1.8);
      if(ex==1){if(move<0.002);else(background(0);move=0;)}
      if(ex==2){if(oneloop==0){background(255);oneloop=1;}}
      circle2D(100,0,0,360);
      makeBezier(0,360,100,0));
  void Bezier_move_change3D(float x0,float y0,float z0,float x1,float y1,
    float z1,float x2,float y2,float z2,float x3,float y3,float z3,float
    x4,float y4,float z4,float x5,float y5,float z5,float k,float R,
    float G,float B,float m,stroke str){
    float i,t=0,u0,v0,u1,v1,u2,v2,w0,v1,w2,u,v,w;
    i=k;
    u0=(1-i)*x0+i*x3;u1=(1-i)*x1+i*x4;u2=(1-i)*x2+i*x5;
    v0=(1-i)*y0+i*y3;v1=(1-i)*y1+i*y4;v2=(1-i)*y2+i*y5;
    w0=(1-i)*z0+i*z3;w1=(1-i)*z1+i*z4;w2=(1-i)*z2+i*z5;
    if(m==0.0){for(t=0.01;t<0.99;t+=0.005){
      u=t*t+u0*2*t*(1-t)+u1*(1-t)*(1-t)*u2;t=t*v0+2*t*(1-t)*v1+(1-t)*(1-t)*u1
      }*2;v=w*t+w0*2*t*(1-t)*v1*(1-t)*(1-t)*w2;
    point3D_RGB(u,v,w,R,G,B,str);}
    else{for(t=0.01;t<1.01;t+=0.005){
      u=t*t+u0*2*t*(1-t)*u1*(1-t)*(1-t)*u2;v=t*t*v0+2*t*(1-t)*v1+(1-t)*(1-t)*u1
      }*2;w=t*t+w0*2*t*(1-t)*w1*(1-t)*(1-t)*w2;
    point3D_RGB(u,v,w,R,G,B,str);}}
  void Bezier_line3D(float x0,float y0,float z0,float x1,float y1,float z1,
    float x2,float y2,float z2,float x3,float y3,float z3,float x4,float
    y4,float z4,float x5,float y5,float z5){
    float z,v,d,e,t=0,u,v;
    for(t=0;t<1;t+=0.2){
      z=t*t*x0+2*t*(1-t)*x1*(1-t)*x2;t=t*x0+2*t*(1-t)*y1+(1-t)*(1-t)*y2;
      y2=t*t+x0*2*t*(1-t)*x1*(1-t)*(1-t)*z2;
      u=t*t*x3+2*t*(1-t)*x4*(1-t)*(1-t)*x5;v=t*t*y3+2*t*(1-t)*y4+(1-t)*(1-t)*y5;
      z5=t*t+z3*2*t*(1-t)*z4*(1-t)*z5;w=t*t+z2*(1-t)*z1*(1-t)*z6;
      strokeWeight(2);line3D_RGB(z,w,d,u,v,e,149,3,255);}
  void Bezier_move3D(float x0,float y0,float z0,float x1,float y1,float z1,
    float x2,float y2,float z2,float x3,float y3,float z3,float x4,float
    y4,float z4,float x5,float y5,float z5){
    float i,t=0,u0,v0,u1,v1,u2,v2,w0,v1,w2,u,v,w;
    for(i=0;1;i<0.9;i+=0.1){
      u0=(1-i)*x0+i*x3;u1=(1-i)*x1+i*x4;u2=(1-i)*x2+i*x5;
      v0=(1-i)*y0+i*y3;v1=(1-i)*y1+i*y4;v2=(1-i)*y2+i*y5;
      w0=(1-i)*z0+i*z3;w1=(1-i)*z1+i*z4;w2=(1-i)*z2+i*z5;
      for(t=0;t<1;t+=0.05){
        u=t*t+u0*2*t*(1-t)*u1*(1-t)*(1-t)*u2;t=t*v0+2*t*(1-t)*v1+(1-t)*(1-t)*u1
        }*2;v=w*t+w0*2*t*(1-t)*v1*(1-t)*(1-t)*w2;
      stroke(28,234,109);strokeWeight(10);point3D_RGB(u,v,w,3,255,101,5);
    }
  void makeBezier(float start,float last,float r,float n){
    goBezier(start,(start+last)/2,0,r,n);
    goBezier((start+last)/2,2,1,r,n);
    if(M[int(n)]>app){
      nogeoBezier(start,(start+last)/2,0,r,n);
      nogeoBezier((start+last)/2,last,1,r,n);
      makeBezier(start,(start+last)/2,r,n+1);
      makeBezier((start+last)/2,2,r,n+1);}
    void nogeoBezier(float start,float last,float r,float num){
      float gammamax,gammahalfx,gamma0x,betalix,betahalfx,beta0x;
      float gammamax,gammahalfy,gamma0y,betalix,betahalfy,beta0y;
      gamma0x=500+r*cos(radians(start));gamma0y=400+r*sin(radians(start));
      gammamax=500+r*cos(radians(last));gamma1y=400+r*sin(radians(last));
      gammahalfx=500+r*cos(radians((start+last)/2));gammahalfy=400+r*sin(radians((start+last)/2));
      betax0=500+r*cos(radians(start));beta0y=400+r*sin(radians(start));
      betax1=500+r*cos(radians(last));beta1y=400+r*sin(radians(last));
      betahalfx=2*gammahalfx-(gamma0x+gamma1x)/2;betahalfy=2*gammahalfy-(gamma0y+gamma1y)/2;
      noBezier(beta0x,beta0y,betahalfx,betahalfy,beta1x,beta1y);
      void noBezier(float x0,float y0,float x1,float y1,float x2,float y2){
        float z,w,t=0;
        for(t=0;t<1;t+=0.005){
          z=t*t*x0+2*t*(1-t)*x1*(1-t)*(1-t)*x2;t=t*x0+2*t*(1-t)*y1+(1-t)*(1-t)*y2;
          stroke(255);strokeWeight(6);point(z,w);}
        void goBezier(float start,float last,float n,float r,float num){
          float gammamax,gammahalfx,gamma0x,betalix,betahalfx,beta0x;
          float gammamax,gammahalfy,gamma0y,betalix,betahalfy,beta0y;
          gamma0x=500+r*cos(radians(start));gamma0y=400+r*sin(radians(start));
          gammamax=500+r*cos(radians(last));gamma1y=400+r*sin(radians(last));
          gammahalfx=500+r*cos(radians((start+last)/2));gammahalfy=400+r*sin(radians((start+last)/2));
          betax0=500+r*cos(radians(start));beta0y=400+r*sin(radians(start));
          betax1=500+r*cos(radians(last));beta1y=400+r*sin(radians(last));
          betahalfx=2*gammahalfx-(gamma0x+gamma1x)/2;betahalfy=2*gammahalfy-(gamma0y+gamma1y)/2;
          point(betahalfx,betahalfy);
          Bezier2D(beta0x,beta0y,betahalfx,betahalfy,beta1x,beta1y,n);
          Hausdorff(100,start,last,betax0,beta0y,betahalfx,betahalfy,beta1x,beta1y,num);
          void circle2D(float r,float n,float start,float last){
            strokeWeight(5);stroke(0);fill(255);ellipse(500,400,2*r,2*r);
            if(n>0){for(float i=0;i<n;i++){
              strokeWeight(0.1);stroke((229,37,105);point(500+r*cos(radians(start+(last-start)/n*i)),500+r*sin(radians(start+(last-start)/n*i))))}}}
        void Bezier2D(float x0,float y0,float x1,float y1,float x2,float y2,float
        num){
          float z,w,t=0;
          for(t=0;t<1;t+=0.005){
            z=t*t*x0+2*t*(1-t)*x1*(1-t)*(1-t)*x2;t=t*x0+2*t*(1-t)*y1+(1-t)*(1-t)*y2;
            stroke(255,0);strokeWeight(5);point(z,w);}
          void Hausdorff(float r,float start,float last,float x0,float y0,float x1,
          float y1,float x2,float y2,float num){
            float t=start,d1=(last-start)/10,k=0,d2=0.1;
            int i,j;
            for(i=0;i<10;i++)k=d2*i;
            for(j=0;j<10;j++){
              H[i][j]=dist(500+r*cos(radians(t)),400+r*sin(radians(t)),x0*(1-k)+k*x1*(1-k)+2*k*(1-k)*x1+k*x2,y0*(1-k)+(1-k)+2*k*(1-k)*y1+k*y2);t+=d1;
              if(min>H[i][j])min=H[i][j];}
            t=start;if(maxmin<min){maxmin=min;}min=100000;
            M[int(num)]=maxmin;maxmin=0;
            stroke(255);fill(255);rect(200,700,900,900);
            stroke(0);fill(0);textSize(30);text("Hausdorff distance=",300,800);
            text(M[int(num)],600,800);}
        void coordinate3D(){
          background(0);
          line3D_RGB(-300,0,0,300,0,0,255,255,255);
          line3D_RGB(0,-300,0,0,300,0,255,255,255);
          line3D_RGB(0,0,-300,0,0,300,255,255,255);
          textSize(20);stroke(255);fill(255,255,255);
          check3D(310,0,0);text("#x",500+A[0],500+A[1]);check3D(-310,0,0);text("-x"
          ,500+A[0],500+A[1]);
          check3D(0,310,0);text("#y",500+A[0],500+A[1]);check3D(0,-310,0);text("-y"
          ,500+A[0],500+A[1]);
          check3D(0,0,310);text("#z",500+A[0],500+A[1]);check3D(0,0,-310);text("-z"
          ,500+A[0],500+A[1]);
          strokeWeight(5);stroke(255);
          textSize(20);text("pi=",85,100);text("ceta=",85,130);
          text(pi,120,100);text(ceta,140,130);}
        void point3D(float x,float y,float z){
          stroke(255);
          A[0]*=cos(radians(ceta))-z*sin(radians(ceta));
          A[1]*=cos(radians(pi))-x*sin(radians(pi))*sin(radians(ceta))-z*cos(radians(ceta));
          strokeWeight(10);stroke(255);point(500+A[0],500+A[1]);
          void point3D_RGB(float x,float y,float z,float R,float G,float B,float st){
            stroke(255);
            A[0]*=cos(radians(ceta))-z*sin(radians(ceta));
            A[1]*=cos(radians(pi))-x*sin(radians(pi))*sin(radians(ceta))-z*cos(radians(ceta));
            A[2]*=cos(radians(ceta));
            A[1]*=cos(radians(pi))-x*sin(radians(pi))*sin(radians(ceta))-z*cos(radians(ceta));
            A[0]*=cos(radians(ceta));
            A[1]*=cos(radians(pi))-x*sin(radians(pi))*sin(radians(ceta))-z*cos(radians(ceta));
          }
        
```

```

139  strokeWeight(st);stroke(R,G,B);point(500+A[0],500+A[1]);}      152
140 void line3D_RGB(float x,float y,float z,float w,float m,float n,float R,   152
141   float G,float B){                                                 153
142   stroke(255);                                                 153
143   A[2]=-x*cos(radians(ceta))-z*sin(radians(ceta));           154
144   A[3]=y*cos(radians(pi))-x*sin(radians(pi))*sin(radians(ceta))-z*cos( 155
145     radians(ceta))*sin(radians(pi));                           156
146   A[4]=-w*cos(radians(ceta))-n*sin(radians(ceta));           156
147   A[5]=-m*cos(radians(pi))-w*sin(radians(pi))*sin(radians(ceta))-n*cos( 158
148     radians(ceta))*sin(radians(pi));                           159
149   strokeWeight(5);stroke(R,G,B);line(500+A[2],500+A[3],500+A[4],500+A[5]); 159
150   void check3D(float x,float y,float z){                      160
151     stroke(255);                                                 160
152     A[0]=-x*cos(radians(ceta))-z*sin(radians(ceta));           161
153     A[1]=y*cos(radians(pi))-x*sin(radians(pi))*sin(radians(ceta))-z*cos( 162
154       radians(ceta))*sin(radians(pi));                           163
155   void Bezier3D(float x0,float y0,float z0,float x1,float y1,float z1,float

```

III.3 3D model approximation program

다음은 3D 구면을 근사하는데 필요한 부분 Section II.2.3 을 구현한 프로세싱의 java 기반 코드이다. 이외에도 저장형식을 불러오기 위한 컴파일러, 함수 찾는 기능, 파일을 저장하는 기능, 불러오는 기능등을 넣은 것이다. 이 코드만 있으면 되는 것이 아닌 보조 텍스트 파일과 그림파일이 필요하다. 하지만 대략적인 구현 방식만 보여주기 위한 것이고 보조 파일의 내용이 너무 많은 관계로 보조 텍스트 파일의 내용을 쓰지는 않겠다.

이 코드를 보면 모두 프로세싱의 java 기반 코딩 프로그램에서 한거지만 java, 즉, 객체 지향적 코딩에 익숙하지 않아 벡터 행렬 객체이외에는 c언어 형식으로 써진 것을 확인 할 수 있다. 이를 인지하고 객체 지향적으로 다시 짠다면 랜더링 시간은 물론이고 전반적인 연구의 결과물도 몇배 좋아질 것이라 생각된다.

```

breakatwhitespace
1 int count=0, count2=0, oneloop1=0, oneloop2=0, oneloop3=0, oneloop4=0, oneloop5
    =0, oneloop6=0, oneloop7=0, row0=0, row1=0, Siz=1, ax=0, ay=0, az=64; 71
    //sphere Siz=1,ax=0,ay=0,az=64; //Heart Siz=5,ax=0,ay=0,az=0 72
2 int RMaterial=128, GMaterial=128, colortexits=-1, windoww=0,
    consolcheck=0, coordinateTRUE=1, Xcodelength=1, searchlength=0, itis=2,
    consoltemp=0; 73
3 int winSoutlength=1,winLength=0,N=10000,ex=0,fend=0,showend=0;
4 int [] [] [] index=new int [100000] [6] [4];
5 float[] A = new float[6]; 74
6 float[] fac=new float [15];
7 float[] Tf=new float [5]; 75
8 float[] Ka=new float [5];
9 float[] Kd=new float [5];
10 float[] Ks=new float [5];
11 float[] Ke=new float [5];
12 float[] ray = new float[6];
13 float[] Acir = new float[6];
14 float[] J3D = new float [6];
15 float[] u = new float [10000000];
16 float[] v = new float [10000000];
17 float[] vsh = new float [10000000];
18 float[] [] com=new float [21] [21];
19 float[] [] obj=new float [100000] [3];
20 float[] [] vervec=new float [100000] [3];
21 float[] [] Drayshow = new float [10] [3];
22 float[] [] new float [10000000] [9];
23 float[] [] [] com2=new float [10] [10] [10];
24 float[] [] [] Snew=float [10] [10] [3];
25 float[] [] [] SRGB=new float [10] [10] [3];
26 float[] [] [] T=new float [5] [5] [3];
27 float[] [] [] show=new float [10] [1000000] [3];
28 float ceta, pi, size=1;
29 float pi_barx=50, pi_bary=83, ceta_barx=50, size_barx=50,
    size_bary=200,RMaterial=50, RMaterialy=375, GMaterialx=100, GMaterialy
    =375, EMaterialx=375; 91
30 float lightx=275, lightxy=325, lightyy=275, lightzz=325, lightzx=375,
    lightzy=325, lightcolorRx=50, lightcolorLy=50, lightcolorGx=50,
    lightcolorLy=650, lightcolorBx=50, lightcolorBy=700; 92
31 float lightpowerx=50, lightpowery=750, lightpowerz=0, coordinateBW=1, lightx
    =128, lighty=128, lightz=128, lightcolorR=255, lightcolorG=255,
    lightcolorB=255, error=256; 93
32 float Ns, Nt, Dr, illum;
33 float zmm=200,zmx=200,ep=0.5; 98
34 String strm, ans=""; //str==> search function string ans== found result
    return 100
35 String[] data= new String [1000000];String[] data2= new String [1000000];//102
    data,count=bzffile data2, count2=vertexfile
36 String[] win4= new String [10000];String[] win5= new String [10000];String[] 104
    [] winSout= new String [10000];
37 PImage img1,img2,img3,img4,img5,img6,img7,img8,img9,img10;
38 vector<PImage> p=new vector<1000000>; vector< [] > b = new vector[3] [3];
39 ArrayStack Ast = new ArrayStack(1000000);
40 ArrayStack Bst = new ArrayStack(1000000);
41 ArrayStack Bstrange = new ArrayStack(1000000); 106
42 void setup() {size(1500, 800);background(0);imageIn();}
43 parseFile(); //obj
44 /*parsefile2(); //mtl
45 }
46 void draw() {if (oneloop1==0){setupcheck();oneloop1=1;}windowcheck();} 108
47 void windowcheck(){
48 if (windowup==0){(Bst.clear());image(img1, 0, 0, 1500, 800);
49 image(img5, 50, 225, 100, 100);textSize(20);fill(0);text("preview", 65, 350);}109
50 if ((mouseButton==LEFT)&&(mouseX>=50)&&(mouseX<=150)&&(mouseY>=225)&&
    (mouseY<=325) {windowup=1;} 110
51 image(img5, 50, 50, 100, 100);textSize(20);fill(0);text("retrieve", 65, 157);111
52 if ((mouseButton==LEFT)&&(mouseX>=50)&&(mouseX<=150)&&(mouseY>=50)&&
    (mouseY<=150) {windowup=2;}; 112
53 image(img5, 50, 400, 100, 100);textSize(20);fill(0);text("save", 80, 525); 113
54 if ((mouseButton==LEFT)&&(mouseX>=50)&&(mouseX<=150)&&(mouseY>=400)&&
    (mouseY<=500) {windowup=3;}; 114
55 image(img5, 200, 50, 100, 100);textSize(20);fill(0);text("compression", 190, 116
    175); 115
56 if ((mouseButton==LEFT)&&(mouseX>=200)&&(mouseX<=300)&&(mouseY>=50)&&
    (mouseY<=150) {windowup=4;}; 117
57 image(img5, 200, 225, 100, 100);textSize(20);fill(0);text("Xcode", 220, 350);118
58 if ((mouseButton==LEFT)&&(mouseX>=200)&&(mouseX<=300)&&(mouseY>=225)&&
    (mouseY<=325) {windowup=5;}; 119
59 image(img4, 50, 575, 100, 100);textSize(20);fill(0);text("option", 70, 700);120
60 if ((mouseButton==LEFT)&&(mouseX>=50)&&(mouseX<=150)&&(mouseY>=575)&&
    (mouseY<=675) {windowup=6;}; 121
61 image(img5, 200, 400, 100, 100);textSize(20);fill(0);text("function", 210, 122
    525); 122
62 if ((mouseButton==LEFT)&&(mouseX>=200)&&(mouseX<=300)&&(mouseY>=400)&&
    (mouseY<=500) {windowup=7;}; 123
63 }
64 if (windowup==1){/preview 127
65 image(img9, 1425, 725, 50, 50);strokeWeight(5);
66 if ((mouseButton==LEFT)&&(mouseX>=1425)&&(mouseY>=725)
    &&(mouseY<=775) {windowup=0;}//Home Button 129
67 if (mouseButton==LEFT) {setangle(0);makecoordinate();}make_bar();rebar();
68 stroke(#FF05FC);fill(#FF05FC); rect(250, 20, 90, 20);fill(0);stroke(0);
    textSize(20);text("retrieve", 250, 38);strokeWeight(5); 130
69 if ((mouseX>=250)&&(mouseY<=340)&&(mouseY>=20)&&(mouseY<=40) &&
    (mouseY>=150) {windowup=1;}; 131
70 BufferedReader codein = createReader("data\\textfile\\codein.txt");
    for (int i=0; i<10000; i++) {win[i]="";}

```

```

135     String line = null;int linecnt=0;
136     try {while ((line = codein.readLine()) != null) {/*println(line);*/}
137         if (linecnt!=0) {win5[linecnt]=line;}linecnt++;
138     codein.close();}208
139     catch (IOException e){e.printStackTrace();}
140     if ((mouseX>=250)&&(mouseX<=340)&&(mouseY>=20)&&(mouseY<=40)) {if(211
141         mouseButton==LEFT){if ((oneloop4==0){oneloop4=0;Bst.clear();consoleQ13
142         ;oneloop4++;} else {oneloop4=0;}}212
143         image(img9, 1425, 725, 50, 50);213
144         if ((mouseButton==LEFT)&&(mouseX>=1425)&&(mouseX<=1475)&&(mouseY>=725)214
145             &&(mouseY<=775) {windowup=0;}}215
146     }216
147     if (windowup==6) {background(255);//option217
148     textSize(20);fill(0);text("coordinate3D",150,200);218
149     madeswitch1(300,183,255,0,0,255,0,80,20);219
150     madeswitch2(300,233,255,0,0,0,255,0,80,20);220
151     fill(120);rect(0,0,100,800);221
152     image(img4,0,100,100);222
153     textSize(30);fill(0);text("Option", 150, 50);223
154     image(img9, 1425, 725, 50, 50);224
155     if ((mouseButton==LEFT)&&(mouseX>=1425)&&(mouseX<=1475)&&(mouseY>=725)225
156         &&(mouseY<=775) {windowup=0;}}226
157     }227
158     if (windowup==7) {background(255);fill(0);//function228
159     rect(0,0,100,800);strokeWeight(1);229
160     image(img6,0,0,100,100);image(img10,250,10,1000,80);230
161     fill(255);strokeWeight(5);rect(260,200,1150,500);231
162     showstring(270,60);232
163     textSize(20);text("Delete? -> clear enter? -> search", 270, 110);233
164     textSize(20);fill(255,0,0);text("NO SHIFT, NO CAPITAL!!",610,110);fill(0);234
165     ;235
166     image(img9, 1425, 725, 50, 50);236
167     if ((mouseButton==LEFT)&&(mouseX>=1425)&&(mouseX<=1475)&&(mouseY>=725)237
168         &&(mouseY<=775) {windowup=0;}}238
169 }239
170 public class vector {240
171     private float[] vec=new float [3];241
172     private float x;242
173     private float y;243
174     private float z;244
175     public vector() {this.x = 0;this.y = 0;this.z=0;this.vec[0]=0;this.vec[1]=0;this.vec[2]=0;}245
176     public vector(float x1,float y1,float z1) {this.x = x1;this.y = y1;this.z=z1;246
177         .z=z1;this.vec[0]=x;this.vec[1]=y;this.vec[2]=z;}247
178     public float norm(vector v) {return sqrt((v.x*v.x+v.y*v.y+v.z*v.z));}248
179     public void displayvector() {System.out.println(this);}249
180     public vector BS(float u, float v){250
181     vector BSu=new vector();251
182     for(int i=0; i<3; i++) for(int j=0; j<3; j++) BS=plus(BS,mul(b[i][j],B252
183         , u)*B[j, v]);return BS;253
184     public vector BSu(float u, float v){254
185     vector BSv=new vector();255
186     for(int i=0; i<3; i++) for(int j=0; j<3; j++) BSu=plus(BSu,mul(b[i][j],B256
187         , u)*B[j, v]);return BSu;257
188     public vector zerovec() {258
189         vector temp = new vector(0,0,0);259
190         return temp;}260
191     public float vectori(int i){return this.vec[i];}261
192     public vector(float[] temp) {this.vec[0] = temp[0];this.vec[1] = temp262
193         [1];this.vec[2] = temp[2];}263
194     public vector plus(vector A, vector B) {float[] temp = this.zerovec().264
195         getvector();265
196         for (int i = 0; i < 3; i++) {temp[i] = A.vectori(i) + B.vec[i];}266
197         return new vector(temp);}267
198     public vector minus(vector A, vector B) {float[] temp = zerovec().268
199         getvector();269
200     public vector plus(vector A, vector B) {270
201         float[] temp = new float[3];float[] tempA = A.getvector();float[]271
202         tempB = B.getvector();272
203         for (int i = 0; i < 3; i++){temp[i] = tempA[i] + tempB[i];}273
204         vector ans = new vector(temp[0],temp[1],temp[2]);274
205         return ans;}275
206     public vector minus(vector A, vector B) {276
207         float[] temp = new float[3];float[] tempA = A.getvector();float[]277
208         tempB = B.getvector();278
209         for (int i = 0; i < 3; i++){temp[i] = tempA[i] - tempB[i];}279
210         public vector mul(vector A, float c) {280
211             float[] temp = new float[3];float[] tempA = A.getvector();float[]281
212             tempA[0]=A.getvector()[0];tempA[1]=A.getvector()[1];
213             tempA[2]=A.getvector()[2];
214             for (int i = 0; i < 3; i++){temp[i] = tempA[i]*c;}282
215             vector ans = new vector(temp[0],temp[1],temp[2]);
216             return ans;}283
217         public vector BS(float u, float v){284
218             vector BS=new vector();
219             for(int i=0; i<3; i++) for(int j=0; j<3; j++) BS=plus(BS,mul(b[i][j],B285
220                 , u)*B[j, v]));return BS;286
221         public vector BSu(float u, float v){287
222             vector BSu=new vector();
223             for(int i=0; i<3; i++) for(int j=0; j<3; j++) BSu=plus(BSu,mul(b[i][j],B288
224                 , u)*B[j, v]));return BSu;289
225         public vector BSv(float u, float v){290
226             vector BSv=new vector();
227             for(int i=0; i<3; i++) for(int j=0; j<3; j++) BSv=plus(BSv,mul(b[i][j],B291
228                 , u)*B[j, v]));return BSv;292
229             public float norm(vector v) {return sqrt(v.x*v.x+v.y*v.y+v.z*v.z);}293
230             float B(int i, float t){if(i==0) {return sqrt((t-1)*(1-t));}else if(i==1)
231                 {return 2*t*(1-t);}}else {return t*t;}}
232             float Bt(int i, float t){if(i==0) {return 2*(t-1);}}else if(i==1) {return
233                 2*(1-2*t);}}else {return 2*t;}}
234             public class ArrayStack {294
235                 private int top;295
236                 private int stackSize;297
237                 private float stackArr[][];
238                 public ArrayStack(int stackSize) {298
239                     top = -1;
240                     this.stackSize = stackSize;
241                     stackArr = new float[this.stackSize][3];
242                     public boolean isEmpty() {return (top == -1);}
243                     public boolean isFull() {return (top == this.stackSize-1);}
244                     public int arraysize() {return top;}
245                     public float iox(int k) {return stackArr[k][0];}
246                     public float iy(int k) {return stackArr[k][1];}
247                     public float iz(int k) {return stackArr[k][2];}
248                     public void push(float x, float y, float z) {299
249                         if (isFull()) /*println("full!");*/
250                             else {top++;stackArr[top][0] = x;stackArr[top][1] = y;stackArr[top][2]
251                                 = z;}}
252                     public void pop() {300
253                         if (isEmpty()) /*println("Deleting fail! Stack is empty!");*/
254                             else /*println("Deleted Item : " + stackArr[top][0] + " " + stackArr[top]
255                                 [1]+ " "+stackArr[top][2]);*/top--;}
256                     public void clear() {301
257                         if (isEmpty()) /*println("Stack is already empty!");*/
258                             else {top = -1;stackArr = new float[this.stackSize][3];/*println("Stack
259                                 is clear!");*/}
260                     public void printStack() {302
261                         if (isEmpty()) /*println("Stack is empty!");*/
262                             else /*print("Bst elements : ");*/
263                             for (int i=0; i<=top; i++) /*println(stackArr[i][0] + " "+stackArr[i]
264                                 [1]+ " "+stackArr[i][2]+ " ");*/point3D_RGB(stackArr[i][0],stackArr[i]
265                                 [1],stackArr[i][2],255,0,0,2,size);
266                             println();}}
267                     float alpha=3;
268                     void go(){307
269                         windclear();
270                         long s=System.currentTimeMillis();
271                         println("alpha:",alpha);
272                         win4push("alpha:"+alpha);
273                         Float.toString(alpha));
274                         println("start approximation"); println("0%"); win4push("0%");
275                         com(zmn+5,0,0,90); println("12.5%"); win4push("12.5%");
276                         com(zmn+5,0,90,180); println("25%"); win4push("25%");
277                         com(zmn+5,0,180,270); println("37.5%"); win4push("37.5%");
278                         com(zmn+5,0,270,360); println("50%"); win4push("50%");
279                         com(0,zmx-5,0,90); println("62.5%"); win4push("62.5%");
280                         com(0,zmx-5,90,180); println("75%"); win4push("75%");
281                         com(0,zmx-5,180,270); println("88.5%"); win4push("88.5%");
282                         com(0,zmx-5,270,360); println("100%"); win4push("100%");
283                         long e=System.currentTimeMillis();
284                         print("finished",e,"ms");
285                         Long.toString(e)+"ms");
286                         win4push("finished"+}
287                     void com(float zmin,float zmax,float anglemin,float anglemax){308
288                         if (anglemax<0)anglemax+=360;if (anglemin<0)anglemin+=360;
289                         if (anglemax>zmax)anglemax=zmax;if (anglemin>zmin)anglemin=zmin;
290                         b[0][0]=new vector(); b[2][0]=new vector(); b[0][2]=new vector();
291                         b[2][2]=new vector(); b[1][0]=new vector(); b[0][1]=new vector();
292                         b[2][1]=new vector(); b[1][2]=new vector(); b[1][1]=new vector();
293                         Bstrange.clear();rangein(zmin,zmax,anglemin,anglemax);if (Bstrange.
294                             arraysize()>0){return;}
295                         float dismax1=0.1,dismax2=0.1,dismax3=0.1,dismax4=0.1;
296                         for(int i=0;i<fend;i++){
297                             ray_triangle(0,0,zmin,zmax,200,zmin,anglemin,f[i][0],f[i][1],f[i][2],f
298                               [i][3],f[i][4],f[i][5],f[i][6],f[i][7],f[i][8]);
299                         }
300                     }

```

```

286 rayshow[0][0]=ray[0];rayshow[0][1]=ray[1];rayshow[0][2]=ray[2];rayshow 344
287 [1][0]*ray[3];rayshow[1][1]=ray[4];rayshow[1][2]=ray[5];
288 checkcirto3D(r3D[0],r3D[1],r3D[2]); 345
289 if((r3D[0]==0 && r3D[1]==0) || Acir[1]<radians(angleremin) || Acir[1]>
radians(angleremax) || Acir[2]>zmin ){} 346
290 else if(dismax1<sqrt(r3D[0]*r3D[0]+r3D[1]*r3D[1]) && !(f[i][1][2]<zmin &&f[1][5]<zmin &&f[i][8]<zmin) && !(f[i][2]>zmin &&f[i][1][5]>zmin &&f[i][8]>zmin){} 348
291 dismax1=sqrt(r3D[0]*r3D[0]+r3D[1]*r3D[1]);b[0][0]=new vector(r3D[0],r3D[1],r3D[2]); 349
292 ray_triangle(0,0,zmin,zmax,200,zmin,angleremax,f[i][0],f[i][1],f[i][2],f[i][3],f[i][4],f[i][5],f[i][6],f[i][7],f[i][8]); 350
293 rayshow[2][0]=ray[0];rayshow[2][1]=ray[1];rayshow[2][2]=ray[2];rayshow 351
[3][0]=ray[3];rayshow[3][1]=ray[4];rayshow[3][2]=ray[5];
294 checkcirto3D(r3D[0],r3D[1],r3D[2]); 352
295 if((r3D[0]==0 && r3D[1]==0) || Acir[1]<radians(angleremin) || Acir[1]>
radians(angleremax) || Acir[2]>zmin ){} 353
296 else if(dismax2<sqrt(r3D[0]*r3D[0]+r3D[1]*r3D[1]) && !(f[i][1][2]<zmin &&f[1][5]<zmin &&f[i][8]<zmin) && !(f[i][2]>zmin &&f[i][5]<zmin &&f[i][8]>zmin){} 356
297 dismax2=sqrt(r3D[0]*r3D[0]+r3D[1]*r3D[1]);b[2][0]=new vector(r3D[0],r3D[57][1],r3D[2]); 358
298 ray_triangle(0,0,zmin,zmax,200,zmax,angleremin,f[i][0],f[i][1],f[i][2],f[i][3],f[i][4],f[i][5],f[i][6],f[i][7],f[i][8]); 360
300 rayshow[4][0]=ray[0];rayshow[4][1]=ray[1];rayshow[4][2]=ray[2];rayshow 361
[5][0]=ray[3];rayshow[5][1]=ray[4];rayshow[5][2]=ray[5];
301 checkcirto3D(r3D[0],r3D[1],r3D[2]); 362
302 if((r3D[0]==0 && r3D[1]==0) || Acir[1]<radians(angleremin) || Acir[1]>
radians(angleremax) || Acir[2]>zmax ){} 363
303 else if(dismax3<sqrt(r3D[0]*r3D[0]+r3D[1]*r3D[1]) && !(f[i][1][2]<zmax &&f[1][5]<zmax &&f[i][8]<zmax) && !(f[i][2]>zmax &&f[i][5]>zmax &&f[i][8]>zmax){} 365
304 dismax3=sqrt(r3D[0]*r3D[0]+r3D[1]*r3D[1]);b[0][2]=new vector(r3D[0],r3D[66][1],r3D[2]); 367
305 ray_triangle(0,0,zmin,zmax,200,zmax,angleremax,f[i][0],f[i][1],f[i][2],f[i][3],f[i][4],f[i][5],f[i][6],f[i][7],f[i][8]); 368
306 rayshow[6][0]=ray[0];rayshow[6][1]=ray[1];rayshow[6][2]=ray[2];rayshow 369
[7][0]=ray[3];rayshow[7][1]=ray[4];rayshow[7][2]=ray[5];
308 checkcirto3D(r3D[0],r3D[1],r3D[2]); 370
309 if((r3D[0]==0 && r3D[1]==0) || Acir[1]<radians(angleremin) || Acir[1]>
radians(angleremax) || Acir[2]>zmax ){} 371
310 else if(dismax4<sqrt(r3D[0]*r3D[0]+r3D[1]*r3D[1]) && !(f[i][1][2]<zmin &&f[1][5]<zmin &&f[i][8]<zmin) && !(f[i][2]>zmax &&f[i][5]>zmax &&f[i][8]>zmax){} 373
311 dismax4=sqrt(r3D[0]*r3D[0]+r3D[1]*r3D[1]);b[2][2]=new vector(r3D[0],r3D[57][1],r3D[2]); 375
312 } 376
313 377
314 if(sqrt(b[0][0].vectori(0)*b[0][0].vectori(0)+b[0][0].vectori(1)*b[0][0].vectori(1))==0){new vector(0,0,zmin);} 378
315 if(sqrt(b[2][0].vectori(0)*b[2][0].vectori(0)+b[2][0].vectori(1)*b[2][0].vectori(1))==0){new vector(0,0,zmin);} 379
316 if(sqrt(b[0][2].vectori(0)*b[0][2].vectori(0)+b[0][1].vectori(0)*b[0][1].vectori(1))==0){new vector(0,0,zmax);} 380
317 if(sqrt(b[2][2].vectori(0)*b[2][2].vectori(0)+b[2][1].vectori(0)*b[2][1].vectori(1))==0){new vector(0,0,zmax);} 381
318 if(sqrt(b[2][2].vectori(0)*b[2][2].vectori(0)+b[2][1].vectori(1)*b[2][1].vectori(1))==0){b[2][2]=new vector(0,0,zmax);} 382
319 float x1=0,y1=0,z1=0,x2=0,y2=0,z2=0,x3=0,y3=0,z3=0,x4=0,y4=0,z4=0,dis1=0, 383
dis2=0,dis3=0,dis4=0; 384
320 float findz=(zmax-zmin)/5*j+zmin; 385
321 for(int j=1;j<=4;j++){float findz=(zmax-zmin)/5*j+zmin; 386
322 for(int i=0;i<end;i++){ 387
323 ray_triangle(0,0,zmin,zmax,200,findz,angleremin,f[i][0],f[i][1],f[i][2],f[i][3],f[i][4],f[i][5],f[i][6],f[i][7],f[i][8]); 388
324 if(r3D[0]==0 && r3D[1]==0 && r3D[2]==0){continue;} 388
325 rayshow[0][0]=ray[0];rayshow[0][1]=ray[1];rayshow[0][2]=ray[2];rayshow 389
[1][0]=ray[3];rayshow[1][1]=ray[4];rayshow[1][2]=ray[5];
327 checkcirto3D(r3D[0],r3D[1],r3D[2]); 389
328 if(Acir[1]>radians(angleremin) || Acir[1]<=5){continue;} 390
329 show[0][showend][0]=x3*0;show[0][showend][1]=r3D[1];show[0][showend] 391
[2][2]=r3D[2];vsh[showend]=0;showend++;
330 float paz,pay,paz,dx,dy,dz,ansx,ansy,ansz; 391
331 paz=r3D[0]-b[0][0].vectori(0);pay=r3D[1]-b[0][0].vectori(1);paz=r3D[2]- 392
b[0][0].vectori(2); 392
dx=b[0][2].vectori(0)-b[0][0].vectori(0);dy=b[0][2].vectori(1)-b[0][0].vectori(1);dz=b[0][2].vectori(2)-b[0][0].vectori(2); 393
ansx=paz*dy-dz*pay;ansy=paz*dx-dz*pay;ansz=paz*px-dy*pay; 393
if( Acir[2]>findz){}else if((ansx*ansx+ansy*ansy+ansz*ansz)/(dx* 394
dx*dy*dy*dz*dz)}{ 394
dis1=(ansx*ansx+ansy*ansy+ansz*ansz)/(dx*dy*dy*dy*dz*dz);x1=r3D[0]*y1- 395
r3D[1]*z1+r3D[2]*j; 395
show[0][showend][0]=x1;show[0][showend][1]=y1;show[0][showend][2]=z1; 395
vsh[showend]=1;showend++; 396
337 for(int j=1;j<=4;j++){float findz=(zmax-zmin)/5*j+zmin; 397
338 for(int i=0;i<end;i++){ 397
339 ray_triangle(0,0,zmin,zmax,200,findz,angleremax,f[i][0],f[i][1],f[i][2],f[i][3],f[i][4],f[i][5],f[i][6],f[i][7],f[i][8]); 399
340 if(r3D[0]==0 && r3D[1]==0 && r3D[2]==0){continue;} 399
rayshow[0][0]=ray[0];rayshow[0][1]=ray[1];rayshow[0][2]=ray[2];rayshow 400
[1][0]=ray[3];rayshow[1][1]=ray[4];rayshow[1][2]=ray[5];
342 checkcirto3D(r3D[0],r3D[1],r3D[2]); 400

```

```

402     show[0][showend][0]=2*x3-mid3x;show[0][showend][1]=2*y3-mid3y;show[0][450
403     showend][2]=2*x2+3-mid3z;vsh[showend]=3;showend++;
404     b[1][2]= new vector(2*x4-mid4x,2*y4-mid4y,2*z4-mid4z);452
405     show[0][showend][0]=2*x4-mid4x;show[0][showend][1]=2*y4-mid4y;show[0][453
406     showend][2]=2*x4-mid4z;vsh[showend]=3;showend++;
407     N=Bstrange.arraysize();b[1][1]=new vector();454
408     for(int i=0;i<N;i++){p[i]=new vector(Bstrange.stackArr[i][0],Bstrange.455
409     stackArr[i][1],Bstrange.stackArr[i][2]);u[i]=0.5;v[i]=0.5;}456
410     winSout[winSoutLength]="checkrc33(0,"+Float.toString(b[0][0].vectori(0))+457
411     ", "+Float.toString(b[0][0].vectori(1))+", "+Float.toString(b[0][0].458
412     vectori(2))+", "+Float.toString(b[0][1].vectori(0))+", "+Float.459
413     toString(b[0][1].vectori(1))+", "+460
414     Float.toString(b[0][1].vectori(2))+", "+Float.toString(b[0][2].vectori(0))+461
415     ", "+Float.toString(b[0][2].vectori(1))+", "+Float.toString(b[0][2].462
416     vectori(2))+", "+Float.toString(b[1][0].vectori(0))+", "+Float.463
417     toString(b[1][0].vectori(1))+", "+464
418     Float.toString(b[1][0].vectori(2))+", "+Float.toString(b[1][1].vectori(0))+465
419     ", "+Float.toString(b[1][1].vectori(1))+", "+466
420     Float.toString(b[1][1].vectori(2))+", "+Float.toString(b[1][2].vectori(0))+467
421     ", "+Float.toString(b[1][2].vectori(1))+", "+468
422     Float.toString(b[1][2].vectori(2))+", "+Float.toString(b[2][0].vectori(0))+469
423     ", "+Float.toString(b[2][0].vectori(1))+", "+470
424     Float.toString(b[2][0].vectori(2))+", "+Float.toString(b[2][1].vectori(0))+471
425     ", "+Float.toString(b[2][1].vectori(1))+", "+472
426     Float.toString(b[2][1].vectori(2))+", "+Float.toString(b[2][2].vectori(0))+473
427     ", "+Float.toString(b[2][2].vectori(1))+", "+474
428     Float.toString(b[2][2].vectori(2))+", "+winSoutLength++;475
429     Rearrange();476
430     show[0][showend][0]=b[1][1].vectori(0);show[0][showend][1]=b[1][1].477
431     vectori(1);show[0][showend][2]=b[1][1].vectori(2);vsh[showend]=4;
432     showend++;478
433     checkrc33(0,b[0][0].vectori(0),b[0][0].vectori(1),b[0][0].vectori(2),b479
434     [0][1].vectori(0),b[0][1].vectori(1),b[0][1].vectori(2),b[0][2].480
435     vectori(0),b[0][2].vectori(1),b[0][2].vectori(2));481
436     checkrc33(1,b[1][0].vectori(0),b[1][0].vectori(1),b[1][0].vectori(2),b482
437     [1][1].vectori(0),b[1][1].vectori(1),b[1][1].vectori(2),b[1][2].483
438     vectori(0),b[1][2].vectori(1),b[1][2].vectori(2));484
439     checkrc33(2,b[2][0].vectori(0),b[2][0].vectori(1),b[2][0].vectori(2),b485
440     [2][1].vectori(0),b[2][1].vectori(1),b[2][1].vectori(2),b[2][2].486
441     vectori(0),b[2][2].vectori(1),b[2][2].vectori(2));487
442     Ast_Bezier_surface_line_rc33(0.05);488
443     float error,error_function();489
444     Ast.clear();490
445     println("point: ",Bstrange.arraysize(),"error=", error, "range=", zmin,zmax491
446     ,anglemin,anglemax);
447     if(error>alpha){492
448       com(zmin,(zmin+zmax)/2,anglemin,(anglemin+anglemax)/2);493
449       com(zmin,(zmin+zmax)/2,(anglemin+anglemax)/2,anglemax);494
450       com((zmin+zmax)/2,zmax,anglemin,(anglemin+anglemax)/2);495
451       com((zmin+zmax)/2,zmax,(anglemin+anglemax)/2,anglemax);}496
452     else{497
453       winSout[winSoutLength]="checkrc33(0,"+Float.toString(b[0][0].vectori(0))+498
454       ", "+Float.toString(b[0][0].vectori(1))+", "+Float.toString(b[0][0].495
455       vectori(2))+", "+Float.toString(b[0][1].vectori(0))+", "+Float.496
456       toString(b[0][1].vectori(1))+", "+497
457       Float.toString(b[0][1].vectori(2))+", "+Float.toString(b[0][2].vectori(0))+498
458       ", "+Float.toString(b[0][2].vectori(1))+", "+499
459       Float.toString(b[0][2].vectori(2))+", "+winSoutLength++;500
460       winSout[winSoutLength]="checkrc33(2,"+Float.toString(b[2][0].vectori(0))+501
461       ", "+Float.toString(b[2][0].vectori(1))+", "+Float.toString(b[2][0].502
462       vectori(2))+", "+Float.toString(b[2][1].vectori(0))+", "+Float.503
463       toString(b[2][1].vectori(1))+", "+504
464       Float.toString(b[2][1].vectori(2))+", "+Float.toString(b[2][2].vectori(0))+505
465       ", "+Float.toString(b[2][2].vectori(1))+", "+506
466       Float.toString(b[2][2].vectori(2))+", "+winSoutLength++;507
467       winSout[winSoutLength]="checkrc33(2,"+Float.toString(b[2][0].vectori(0))+508
468       ", "+Float.toString(b[2][0].vectori(1))+", "+Float.toString(b[2][0].509
469       vectori(2))+", "+Float.toString(b[2][1].vectori(0))+", "+Float.510
470       toString(b[2][1].vectori(1))+", "+511
471       Float.toString(b[2][1].vectori(2))+", "+Float.toString(b[2][2].vectori(0))+510
472       ", "+Float.toString(b[2][2].vectori(1))+", "+511
473       Float.toString(b[2][2].vectori(2))+", "+winSoutLength++;512
474       winSout[winSoutLength]="beziersurfaceliner33("+Float.toString(255)+"."+513
475       Float.toString(0)+"."+Float.toString(255)+"."+0.0+"."+Float.toString(514
476       (0.05)+"");winSoutLength++;515
477   }
478   void rangeIn(float zmin,float zmax,float anglemin,float anglemax){516
479   for(int i=0;i<row;i++){checkcirto3D(obj[i][0]-ax,obj[i][1]-ay,obj[i][2]-518
480   az);519
481   if(Acir[2]>zmin & Acir[2]<=zmax & Acir[1]>radians(anglemin) & Acir[519
482   [1]<radians(anglemax)){520
483   Bstrange.push(obj[i][0]-ax,obj[i][1]-ay,obj[i][2]-az);}}521
484   float error_function(){522
485   float error=le9;
486   if(min(Bstrange.arraysize(),Ast.arraysize())<0){return 1;}
487   for(int i=0;i<min(Bstrange.arraysize(),Ast.arraysize());i++){
488   vector err=new vector(Ast.ix(i)-Bstrange.ix(i),Ast.iy(i)-Bstrange.y(i)
489   ,Ast.iz(i)-Bstrange.iz(i));
490   if(norm(err)<error){error=norm(err);}
491   return error;
492   }
493   void Find_b11(){
494   vector r= new vector(); float s=0.01;
495   for(int i=0; i<N; i++){
496   if(norm(BSv(u[i], v[i]))<0.001 || norm(BSv(u[i], v[i]))<0.001)
497   continue;
498   vector y= new vector();
499   y=minus(BSv(u[i], v[i]),mul(b[1][1],B(1, u[i])*B(1, v[i])));
500   r=plus(r,mul(minus(p[i],y),B(1, u[i])*B(1, v[i])));
501   s+=B(1, u[i])*B(1, u[i])*B(1, u[i])*B(1, v[i]);
502   b[1][1]=mul(r,s/);
503   }
504   void Rearrange(){
505   for(int i=0; i<N; i++){
506   if(norm(BSv(u[i], v[i]))<0.001 || norm(BSv(u[i], v[i]))<0.001)
507   continue;
508   float ut=[i]-ep*(BSv(u[i], v[i]).vectori(0)*(minus(BSv(u[i], v[i]),p[i])
509   .vectori(0)+BSv(u[i], v[i]).vectori(1)*(minus(BSv(u[i], v[i]),p[i])
510   .vectori(1)+BSv(u[i], v[i]).vectori(2)*(minus(BSv(u[i], v[i]),p[i])
511   .vectori(2)+BSv(u[i], v[i]).norm(BSv(u[i], v[i]))));
512   float vt=[i]-ep*(BSv(u[i], v[i]).vectori(0)*(minus(BSv(u[i], v[i]),p[i]
513   .vectori(0)+BSv(u[i], v[i]).vectori(1)*(minus(BSv(u[i], v[i]),p[i]
514   .vectori(1)+BSv(u[i], v[i]).vectori(2)*(minus(BSv(u[i], v[i]),p[i]
515   .vectori(2)+BSv(u[i], v[i]).norm(BSv(u[i], v[i]))));
516   u[i]=ut; v[i]=vt;
517   }
518   void Ast_Bezier_surface_line_rc33(float gap){
519   int i,j;ifloat u,v;float[] [] save=new float[1005][1005][3];
520   for(u=0;u<1.001;u+=gap){for(v=0.0;v<1.001;v+=gap){float sumx=0,
521   sumz=0;
522   for(i=0;i<3;i++){for(j=0;j<3;j++){sumx+=com[2][i]*com[2][j]*power(u,i)
523   *power(v,j)*power(1-u,2-i)*power(1-v,2-j)*S[i][j][0];}
524   for(i=0;i<3;i++){for(j=0;j<3;j++){sumy+=com[2][i]*com[2][j]*power(u,i)
525   *power(v,j)*power(1-u,2-i)*power(1-v,2-j)*S[i][j][1];}
526   for(i=0;i<3;i++){for(j=0;j<3;j++){sumz+=com[2][i]*com[2][j]*power(u,i)
527   *power(v,j)*power(1-u,2-i)*power(1-v,2-j)*S[i][j][2];}
528   save[(int)(u*1000)][(int)(v*1000)][0]=sumx;save[(int)(u*1000)][(int)(v*1000)][1]=sumy;save[(int)(u*1000)][(int)(v*1000)][2]=sumz;
529   Ast.push(sumx,sumy,sumz);}}
530   float floattry(String str){
531   float[] subfloat = new float[2];float ans;
532   int now=0;subfloat[0]=0;subfloat[1]=0;String[] newfloat = split(str,' ');
533   for(int i=0;i<2;i++){for(int j=0;j<newfloat[i].length();j++){
534   if(newfloat[i].substring(j,j+1).equals("E")){now=0;}
535   if(newfloat[i].substring(j,j+1).equals("1")){now=1;}
536   if(newfloat[i].substring(j,j+1).equals("2")){now=2;}
537   if(newfloat[i].substring(j,j+1).equals("3")){now=3;}
538   if(newfloat[i].substring(j,j+1).equals("4")){now=4;}
539   if(newfloat[i].substring(j,j+1).equals("5")){now=5;}
540   if(newfloat[i].substring(j,j+1).equals("6")){now=6;}
541   if(newfloat[i].substring(j,j+1).equals("7")){now=7;}
542   if(newfloat[i].substring(j,j+1).equals("8")){now=8;}
543   if(newfloat[i].substring(j,j+1).equals("9")){now=9;}
544   subfloat[i]=(10+subfloat[i]*now);}ans=subfloat[0]+subfloat[1]/pow(10,now);
545   ans=power(10,intry(newfloat[i].substring(j+2,newfloat[i].length())));
546   }return ans;}
547   else{ ans=subfloat[0]+subfloat[1]/pow(10,now);}
548   ans=power(10,intry(newfloat[i].substring(j+1,newfloat[i].length())));
549   }return ans;}
550   if(newfloat[i].substring(j,j+1).equals("0")){now=0;}
551   if(newfloat[i].substring(j,j+1).equals("1")){now=1;}
552   if(newfloat[i].substring(j,j+1).equals("2")){now=2;}
553   if(newfloat[i].substring(j,j+1).equals("3")){now=3;}
554   if(newfloat[i].substring(j,j+1).equals("4")){now=4;}
555   if(newfloat[i].substring(j,j+1).equals("5")){now=5;}
556   if(newfloat[i].substring(j,j+1).equals("6")){now=6;}
557   if(newfloat[i].substring(j,j+1).equals("7")){now=7;}
558   if(newfloat[i].substring(j,j+1).equals("8")){now=8;}
559   if(newfloat[i].substring(j,j+1).equals("9")){now=9;}
560   subfloat[i]=(10+subfloat[i]*now);}ans=subfloat[0]+subfloat[1]/pow(10,newfloat[i].length());return ans;}
561   float floattry2(String line)
562   { float line2;
563   if((line.substring(0,1)).equals("-")){line2=(-1)*floattry(line.substring(1,line.length()-1));
564   }else{line2=floattry(line);}
565   return line2;}
566   int inttry(String str){
567   int ans=0;now=0;String[] newfloat = new String [2];newfloat[0]=str;//
568   printin(str);
569   for(int i=0;i<1;i++){for(int j=0;j<newfloat[i].length();j++){
570   if(newfloat[i].substring(j,j+1).equals("0")){now=0;}
571   if(newfloat[i].substring(j,j+1).equals("1")){now=1;}
572   if(newfloat[i].substring(j,j+1).equals("2")){now=2;}
573   if(newfloat[i].substring(j,j+1).equals("3")){now=3;}
574   if(newfloat[i].substring(j,j+1).equals("4")){now=4;}
575   if(newfloat[i].substring(j,j+1).equals("5")){now=5;}
576   if(newfloat[i].substring(j,j+1).equals("6")){now=6;}
577   if(newfloat[i].substring(j,j+1).equals("7")){now=7;}
578   if(newfloat[i].substring(j,j+1).equals("8")){now=8;}
579   if(newfloat[i].substring(j,j+1).equals("9")){now=9;}
580   }
581   }
582   }

```

```

525 ans*=10;ans+=now;}}return ans;} 595
526 596
527 void cylinder_ray(float x,float y, float zmin,float zmax, float r,float 597
528 float x1,y1,z1,x2,y2,z2; 598
529 x1+=r*cos(radians(angle));y1=y+r*sin(radians(angle));z1=findz; 599
530 z2=x1;y2=y2;z2=findz; 600
531 line3D(x1,y1,z1,x2,y2,z2,size,255,0,0,3); 601
532 /*point3D_RGB(x1,y1,z1,250,0,250,7,size);point3D_RGB(x2,y2,z2 602
533 ,250,0,250,7,size*/; 603
534 ray[0]=x1;ray[1]=y1;ray[2]=z1;ray[3]=x2;ray[4]=y2;ray[5]=z2; 604
535 void ray_triangle(float x,float y, float zmin,float zmax, float r,float 605
536 findz,float angle,float trix1,float triy1,float triz1,float trix2, 606
537 float triy2,float triz2,float trix3,float triy3,float triz3){ 607
538 float x1,y1,z1,x2,y2,z2,dx,dy,dz,x0,y0,z0,t,u,v,e1x,e1y,e2x,e2y,e2z, 608
539 Tx,Ty,Tz,Px,Py,Pz,Qx,Qy,Qz; 609
540 x1+=r*cos(radians(angle));y1+=r*sin(radians(angle));z1=findz;x2=y2+ 610
541 ;z2=findz; 611
542 ray[0]=x;ray[1]=y1;ray[2]=z1;ray[3]=x2;ray[4]=y2;ray[5]=z2; 612
543 dx=(ray[0]-ray[3])*sqrt((ray[3]-ray[0])*(ray[3]-ray[0])+ (ray[4]-ray[1])* 613
544 (ray[4]-ray[1])+ (ray[5]-ray[2])* (ray[5]-ray[2])); 614
545 dy=(ray[1]-ray[4])*sqrt((ray[3]-ray[0])*(ray[3]-ray[0])+ (ray[4]-ray[1])* 615
546 (ray[4]-ray[1])+ (ray[5]-ray[2])* (ray[5]-ray[2])); 616
547 dz=(ray[0]-ray[5])*sqrt((ray[3]-ray[0])*(ray[3]-ray[0])+ (ray[4]-ray[1])* 617
548 (ray[4]-ray[1])+ (ray[5]-ray[2])* (ray[5]-ray[2])); 618
549 x0=y0;y0=z0;findz; 619
550 e1x=trix2-trix1;e1y=triy2-triy1;e1z=triz2-triz1;e2x=trix3-trix1;e2y=triy 620
551 -triy1;e2z=triz3-triz1; 621
552 Tx=x0-trix1;Ty=y0-triy1;Tz=z0-triz1; 622
553 Px=dy*2*x-dz*x2;Py=dx*2*x-dx*z2;Px+dz*e2y-dy*e2x;Qx=Ty*eilz-Tz*eily;Qy*Tz 623
554 *eilx;Tx*eilz;Qz=Tx*eily-Ty*eilx; 624
555 t=(Qx*eilx+Qy*eily+Qz*eilz);(Px*eilx+Py*eily+Pz*eilz); 624
556 *eilx+Py*eily+Pz*eilz);(Px*eilx+Py*eily+Pz*eilz);(Px*eilx+Py*eily+Pz*eilz); 625
557 if((u>0)&&(u<1)&&(v>0)&&(v<1)) {x3D[0]=x0+t*dx;x3D[1]=y0+t*dy;x3D[2]= 626
558 z0+t*dz;*point3D_RGB(x0+t*dx,y0+t*dy,z0+t*dz,0,250,60,10,size); 627
559 ;}*else{r3D[0]=0;r3D[1]=0;r3D[2]=0;} 628
560 void tri_gappoint(float trix1,float triy1,float triz1,float trix2,float 629
561 triy2,float triz2,float trix3,float triy3,float triz3,float gap){ 630
562 float x,y,z; 631
563 for(float u=0;u<1;u+=gap){for(float v=0;v<1;v+=gap){ 632
564 if(u+v>1){continue;} 633
565 x=trix1*u+trix2*v+trix3*(1-u-v); 634
566 y=triy1*u+triy2*v+triy3*(1-u-v); 635
567 z=triz1*u+triz2*v+triz3*(1-u-v); 636
568 point3D_RGB(x,y,z,250,150,0,1,size);}} 637
569 void triangle_3D(float trix1,float triy1,float triz1,float trix2,float 638
570 triy2,float triz2,float trix3,float triy3,float triz3){ 639
571 line3D(trix1,triy1,tri1,tri2,tri2,tri2,size,250,150,0,1); 640
572 line3D(trix2,triy2,tri2,tri2,tri3,tri3,tri3,size,250,150,0,1); 641
573 line3D(trix3,triy3,tri3,tri1,tri1,tri1,tri1,size,250,150,0,1); 642
574 line3D((trix1+trix2)/2,(triy1+triy2)/2,(triz1+triz2)/2,(trix2+trix3)/2,( 643
575 triy2+triy3)/2,(triz2+triz3)/2,size,250,150,0,1); 644
576 line3D((trix2+trix3)/2,(triy2+triy3)/2,(triz2+triz3)/2,(trix3+trix1)/2,( 645
577 triy3+triy1)/2,(triz3+triz1)/2,size,250,150,0,1); 646
578 line3D((trix3+trix1)/2,(triy3+triy1)/2,(triz3+triz1)/2,(trix1+trix2)/2,( 647
579 triy1+triy2)/2,(triz1+triz2)/2,size,250,150,0,1); 648
580 line3D((trix1+trix2)/2,(triy1+triy2)/2,(triz1+triz2)/2,(trix3+trix1)/2,( 649
581 triy3+triy1)/2,(triz3+triz1)/2,size,250,150,0,1);} 650
582 651
583 void cylinder(float x,float y, float zmin,float zmax, float r){ 652
584 float z,w,m,n,l,p; 653
585 for(int i=0;i<360;i++)for(int j=0;j<5;j++){ 654
586 z=r*cos(radians(i));x=w*r*sin(radians(i));y=m=(zmax-zmin)/5*j+zmin; 655
587 point3D_RGB(z,w,m,0,202,250,1,size); 656
588 z=r*cos(radians(i));x=w*r*sin(radians(i))+y;m=zmin;l=r*cos(radians(i)) 657
589 );*x;p=r*sin(radians(i))+y;n=zmax; 660
590 point3D_RGB(z,w,m,0,202,250,1,size);point3D_RGB(l,p,n,0,202,250,1, 661
591 size); 662
592 if(i%45==0){line3D(z,w,m,l,p,n,size,0,202,250,1);}} 663
593 664
594 void checkcirto3D(float x,float y,float z){ 665
595 Acir[0]=sqrt(x*x+y*y); 666
596 if(x>0 && y>0){Acir[1]=atan(y/x);} 667
597 else if(x<0 && y>0){Acir[1]=PI/2+atan(-x/y);} 668
598 else if(x<0 && y<0){Acir[1]=PI+atan(y/x);} 669
599 else if(x>0 && y<0){Acir[1]=3*PI/2+atan(-x/y);} 670
600 Acir[2]=z; 671
601 void makecoordinate(){ 672
602 strokeWeight(5);fill(0); 673
603 if((coordinateTRUE==1){ 674
604 line3D(-.350, 0, .350, 0, 0, size, 255*coordinateBW, 255*coordinateBW 675
605 ,255*coordinateBW, 5); 676
606 line3D(0, -.350, 0, .350, 0, size, 255*coordinateBW, 255*coordinateBW 677
607 ,255*coordinateBW, 5); 678
608 line3D(0, -.350, 0, .350, 0, size, 255*coordinateBW, 255*coordinateBW 679
609 ,255*coordinateBW, 5); 680
610 line3D(0, -.350, 0, .350, 0, size, 255*coordinateBW, 255*coordinateBW 681
611 ,255*coordinateBW, 5); 682
612 textSize(20);stroke(255);fill(255*coordinateBW); 683
613 check3D(360, 0, 0, size); text("x", 950*A[0], 400+A[1]); 684
614 check3D(-360, 0, 0, size); text("x", 950*A[0], 400+A[1]); 685
615 check3D(0, 360, 0, size); text("y", 950*A[0], 400+A[1]); 686
616 check3D(0, -360, 0, size); text("y", 950*A[0], 400+A[1]); 687
617 check3D(0, 0, 360, 0, size); text("z", 950*A[0], 400+A[1]); 688
618 689
619 check3D(0, 0, -360, size); text("-x", 950*A[0], 400+A[1]); 690
620 check3D(0, 0, 360, size); text("-x", 950*A[0], 400+A[1]); 691
621 check3D(0, -360, 0, size); text("-y", 950*A[0], 400+A[1]); 692
622 check3D(0, 0, -360, size); text("-y", 950*A[0], 400+A[1]); 693
623 check3D(0, 0, 0, 360, size); text("-z", 950*A[0], 400+A[1]); 694
624 695
625 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 696
626 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 697
627 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 698
628 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 699
629 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 700
630 701
631 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 702
632 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 703
633 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 704
634 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 705
635 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 706
636 707
637 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 708
638 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 709
639 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 710
640 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 711
641 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 712
642 713
643 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 714
644 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 715
645 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 716
646 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 717
647 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 718
648 719
649 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 720
650 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 721
651 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 722
652 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 723
653 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 724
654 725
655 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 726
656 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 727
657 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 728
658 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 729
659 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 730
660 731
661 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 732
662 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 733
663 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 734
664 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 735
665 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 736
666 737
667 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 738
668 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 739
669 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 740
670 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 741
671 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 742
672 743
673 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 744
674 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 745
675 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 746
676 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 747
677 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 748
678 749
679 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 750
680 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 751
681 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 752
682 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 753
683 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 754
684 755
685 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 756
686 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 757
687 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 758
688 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 759
689 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 760
690 761
691 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 762
692 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 763
693 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 764
694 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 765
695 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 766
696 767
697 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 768
698 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 769
699 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 770
700 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 771
701 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 772
702 703
703 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 773
704 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 774
705 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 775
706 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 776
707 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 777
708 709
709 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 778
710 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 779
711 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 780
712 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 781
713 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 782
714 715
715 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 783
716 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 784
717 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 785
718 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 786
719 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 787
720 721
721 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 788
722 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 789
723 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 790
724 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 791
725 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 792
726 727
727 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 793
728 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 794
729 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 795
730 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 796
731 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 797
732 733
733 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 798
734 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 799
735 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 800
736 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 801
737 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 802
738 803
739 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 804
740 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 805
741 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 806
742 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 807
743 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 808
744 809
745 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 810
746 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 811
747 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 812
748 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 813
749 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 814
750 811
751 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 812
752 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 813
753 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 814
754 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 815
755 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 816
756 817
757 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 818
758 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 819
759 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 820
760 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 821
761 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 822
762 823
763 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 824
764 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 825
765 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 826
766 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 827
767 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 828
768 829
769 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 830
770 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 831
771 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 832
772 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 833
773 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 834
774 835
775 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 836
776 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 837
777 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 838
778 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 839
779 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 840
780 841
781 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 842
782 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 843
783 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 844
784 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 845
785 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 846
786 847
787 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 848
788 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 849
789 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 850
790 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 851
791 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 852
792 853
793 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 854
794 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 855
795 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 856
796 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 857
797 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 858
798 859
799 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 860
800 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 861
801 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 862
802 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 863
803 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 864
804 865
805 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 866
806 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 867
807 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 868
808 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 869
809 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 870
810 871
811 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 872
812 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 873
813 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 874
814 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 875
815 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 876
816 877
817 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 878
818 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 879
819 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 880
820 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 881
821 check3D(0, 0, 0, 360, size); text("y", 950*A[0], 400+A[1]); 882
822 883
823 check3D(0, 0, -360, size); text("z", 950*A[0], 400+A[1]); 884
824 check3D(0, 0, 360, size); text("z", 950*A[0], 400+A[1]); 885
825 check3D(0, -360, 0, size); text("x", 950*A[0], 400+A[1]); 886
826 check3D(0, 0, -360, size); text("x", 950*A[0], 400+A[1]); 887

```

```

    mouseY<y2)&&(colorExits==1)) {
682     GMatrialy=mouseY;GMaterial=255-(int)((GMatrialy-y1)/(y2-y1)*255);} 764
683     strokeWeight(5);} 765
684 void bar6(float x, float y1, float y2) { //EMaterial 766
685     EMatrialx=x; 767
686     fill(0);rect(x-10, y1-10, 20, y2+y1+20); 768
687     fill(100);rect(x-5, y1, 10, y2-y1); 769
688     fill(255, 0, 0);ellipse(EMatrialx, EMatrialy, 15, 15); 770
689     if ((mouseButton==LEFT)&&(mouseX<x+10)&&(mouseX>x-10)&&(mouseY>y1)&& 771
690         mouseY<y2)&&(colorExits==1)) { 772
691     EMatrialy=mouseY;BMaterial=255-(int)((EMatrialy-y1)/(y2-y1)*255);} 773
692     strokeWeight(5);} 773
693 void bar7(float x, float y1, float y2) { //lightx 774
694     lightxx=x; 775
695     fill(0);rect(x-10, y1-10, 20, y2-y1+20); 776
696     fill(100);rect(x-5, y1, 10, y2-y1); 777
697     fill(255, 0, 0);ellipse(lightxx, lightxy, 15, 15); 778
698     if ((mouseButton==LEFT)&&(mouseX<x+10)&&(mouseX>x-10)&&(mouseY>y1)&& 779
699         mouseY>y2)) { 780
700     lightxy=mouseY;lightx=255-(int)((lightxy-y1)/(y2-y1)*512);} 780
701     strokeWeight(5);} 780
702 void bar8(float x, float y1, float y2) { //lighty 781
703     lightyy=x; 782
704     fill(0);rect(x-10, y1-10, 20, y2-y1+20); 783
705     fill(100);rect(x-5, y1, 10, y2-y1); 784
706     fill(255, 0, 0);ellipse(lightyy, lightyy, 15, 15); 785
707     if ((mouseButton==LEFT)&&(mouseX<x+10)&&(mouseX>x-10)&&(mouseY>y1)&& 785
708         mouseY>y2)) { 786
709     lightyy=mouseY;lighty=255-(int)((lightyy-y1)/(y2-y1)*512);} 786
710     strokeWeight(5);} 786
711 void bar9(float x, float y1, float y2) { //lightz 787
712     lightzz=x; 788
713     fill(0);rect(x-10, y1-10, 20, y2-y1+20); 789
714     fill(100);rect(x-5, y1, 10, y2-y1); 790
715     fill(255, 0, 0);ellipse(lightzz, lightzy, 15, 15); 791
716     if ((mouseButton==LEFT)&&(mouseX<x+10)&&(mouseX>x-10)&&(mouseY>y1)&& 792
717         mouseY>y2)) { 792
718     lightzy=mouseY;lightz=255-(int)((lightzy-y1)/(y2-y1)*512);} 793
719     strokeWeight(5);} 793
720 void bar10(float y, float x1, float x2) { //lightcolorR 794
721     lightcolorRy=y; 795
722     fill(0);rect(x1-10, y-10, x2-x1+20, 20); 797
723     fill(100);rect(x1, y-5, x2-x1, 10); 798
724     fill(255, 0, 0);ellipse(lightcolorRx, lightcolorRy, 15, 15); 799
725     if ((mouseButton==LEFT)&&(mouseX<x2)&&(mouseX>x1)&&(mouseY>y-10)&& 800
726         mouseY>y10)) { 801
727     lightcolorRx=mouseX;lightcolorR=(lightcolorRx-x1)/(x2-x1)*255;} 801
728     strokeWeight(5);} 801
729 void bar11(float y, float x1, float x2) { //lightcolorG 802
730     lightcolorGy=y; 802
731     lightcolorGy= 803
732     fill(0);rect(x1-10, y-10, x2-x1+20, 20); 803
733     fill(100);rect(x1, y-5, x2-x1, 10); 804
734     fill(255, 0, 0);ellipse(lightcolorGx, lightcolorGy, 15, 15); 804
735     if ((mouseButton==LEFT)&&(mouseX<x2)&&(mouseX>x1)&&(mouseY>y-10)&& 804
736         mouseY>y10)) { 805
737     lightcolorGx=mouseX;lightcolorG=(lightcolorGx-x1)/(x2-x1)*255;} 805
738     strokeWeight(5);} 805
739 void bar12(float y, float x1, float x2) { //lightcolorB 806
740     lightcolorBy=y; 807
741     fill(0);rect(x1-10, y-10, x2-x1+20, 20); 807
742     fill(100);rect(x1, y-5, x2-x1, 10); 808
743     fill(255, 0, 0);ellipse(lightcolorBx, lightcolorBy, 15, 15); 809
744     if ((mouseButton==LEFT)&&(mouseX<x2)&&(mouseX>x1)&&(mouseY>y-10)&& 809
745         mouseY>y10)) { 810
746     lightcolorBx=mouseX;lightcolorB=(lightcolorBx-x1)/(x2-x1)*255;} 810
747     strokeWeight(5);} 810
748 void bar13(float y, float x1, float x2) { //lightpower 811
749     lightpowery=y; 812
750     fill(0);rect(x1-10, y-10, x2-x1+20, 20); 813
751     fill(100);rect(x1, y-5, x2-x1, 10); 814
752     fill(255, 0, 0);ellipse(lightpowerx, lightpowery, 15, 15); 815
753     if ((mouseButton==LEFT)&&(mouseX<x2)&&(mouseX>x1)&&(mouseY>y-10)&& 815
754         mouseY>y10)) { 816
755     lightpowerx=mouseX;lightpower=(lightpowerx-x1)/(x2-x1)*30;} 816
756     strokeWeight(5);} 816
757 void madeswitch1(float x, float y, int R1, int G1, int B1, int R2, int G2, 817
758     int B2, float dx, float dy) { 817
759     strokeWeight(5); 818
760     fill(255);rect(x, y, dx, dy); 818
761     if ((coordinateTRUE==1) {fill(R2, G2, B2);rect(x, y, dx/2, dy);} 819
762     if ((coordinateTRUE==1) {fill(R1, G1, B1);rect(x+dx/2, y, dx/2, dy);} 820
763     if ((mouseButton==LEFT)&&(mouseX>x+dx)&&(mouseY>y)&& 820
        mouseY<=y+dy)) {coordinateTRUE=-1;} 820
        if ((mouseButton==LEFT)&&(mouseX>x)&&(mouseX<=x+dx/2)&&(mouseY>y)&& 820
            mouseY<=y+dy)) {coordinateTRUE=1;} 820
void madeswitch2(float x, float y, int R1, int G1, int B1, int R2, int G2, 820
    int B2, float dx, float dy) { 820
    strokeWeight(5); 820
    fill(255);rect(x, y, dx, dy); 820
    if ((coordinateBW==1) {fill(R2, G2, B2);rect(x, y, dx/2, dy);} 820
        if ((coordinateBW==1) {fill(R1, G1, B1);rect(x+dx/2, y, dx/2, dy);} 820
        if ((mouseButton==LEFT)&&(mouseX>x+dx/2)&&(mouseX<x+dx)&&(mouseY>y)&& 820
            mouseY<=y+dy)) {coordinateBW=1;} 820
        if ((mouseButton==LEFT)&&(mouseX>x)&&(mouseX<x+dx/2)&&(mouseY>y)&& 820
            mouseY<=y+dy)) {coordinateBW=1;} 820
void setangle() { 820
    background(255-255*coordinateBW);fill(#FFEA00);rect(0, 0, 400, 800); 820
    fill(255*coordinateBW);rect(370, 0, 25, 800);fill(0); 820
    if ((mouseY==400)&&(mouseY>0)&&(mouseY<800)&&(mouseX<1500)) { 820
        ceta=(mouseX-400)/1100.*360.0;pi=mouseY/800.0*360.0; 820
        text("angle", 45, 30);text("phi", 45, 60);text("theta", 45, 120);text(" 820
            size, 45, 180); 820
        fill(255, 0, 0);text("R", 45, 275);fill(0);text("G", 95, 275);fill(0, 0, 820
            255);text("B", 145, 275); 820
        fill(0);text("Light?", 245, 245); 820
        fill(0);text("x", 220, 275);fill(0);text("y", 270, 275);fill(0);text("z", 320, 275); 820
        stroke(#0091FC);fill(#0091FC); rect(250, 20, 90, 20);fill(0);stroke(0); 820
        textSize(20);text("relive", 259, 38);strokeWeight(5); 820
        if ((mouseX==250)&&(mouseX<=340)&&(mouseY>20)&&(mouseY<=40)&& 820
            mouseButton==LEFT)) {Sel.clear();consolcheck=2;} 820
        if ((colorExits==1) {fill(255, 0, 0);text("fixed!", 75, 520);} 820
        fill(0);text("Light(G,B,Power)", 40, 560); 820
        fill(255, 0, 0);text("R", 20, 605);fill(0);text("G", 20, 655);fill(0, 0, 820
            255);text("B", 20, 705);fill(0);text("P", 20, 755); 820
        text(pi, 85, 60);text(ceta, 105, 120);text(size, 100, 180); 820
        text(RMaterial, 30, 485);text(GMaterial, 80, 485);text(BMaterial, 130, 485); 820
        text((int)lightx, 205, 485);text((int)lighty, 255, 485);text((int)lightz, 305, 485); 820
        text((int)lightcolorR, 300, 605);text((int)lightcolorG, 300, 655);text((int)lightcolorB, 300, 705); 820
        text((int)lightpower, 300, 755); 820
        fill(0);stroke(0);} 820
void mouseWheel(MouseEvent event) { 820
    if ((windowup==1){ 820
        if (((mouseX>400)&&(mouseY>0)&&(mouseY<800)&&(mouseX<1500))&&(size 820
            >0.1)&&(size<=1.49)) { 820
            float e = event.getCount()*0.01;size=e;setangle();makecoordinate(); 820
            strokeWeight(5);fill(0);make_bar();rebar();} 820
            void BezierData(float x0,float y0,float z0,float x1,float y1,float 820
                z1,float x2,float y2,float z2, float num, float siz){if(siz==0){size 820
                    ;} 820
                    data[count]="#function name: Bezier3D 11 sources"; count++;data[count]="# 820
                        x0: "+Float.toString(x0); count++;data[count]="#y0: "+Float.toString(y0); 820
                        count++; 820
                        data[count]="#z0: "+Float.toString(z0); count++;data[count]="#x1: "+Float 820
                            .toString(x1); count++;data[count]="#y1: "+Float.toString(y1); count++; 820
                            data[count]="#z1: "+Float.toString(z1); count++; 820
                            data[count]="#x2: "+Float.toString(x2); count++;data[count]="#y2: "+Float 820
                                .toString(y2); count++;data[count]="#z2: "+Float.toString(z2); count++; 820
                                data[count]="#num: "+num; "#num: "+Float.toString(num); count++; 820
                                data[count]="#siz: "+Float.toString(siz); count++;data[count]="# 820
                                    bezier3ddata: point:";count++; 820
                                    float z,w,k,t=0; 820
                                    for (t=0;t<=1.0005){#=t*x*t*x*0.2*t*(1-t)*x1+(1-t)*(1-t)*x2;w=t*t*y0+2*t 820
                                        *(1-t)*y1+(1-t)*y2;k=t*t*x*0.2*t*(1-t)*z1+(1-t)*(1-t)*z2; 820
                                        point3D_RGBData(z,w,k,255,0,0.5,siz);Bpoint3D(z,w,k); 820
                                        data[count]="#bezier3ddata: control point:";count++; 820
                                        point3Dcvedata(x0,y0,z0,siz);count++;point3Dcvedata(x1,y1,z1,siz);count2 820
                                            --;point3Dcvedata(x2,y2,z2,siz);count2--;//control 820
                                            stroke(0);fill(0);} 820
void Bezier_surface_control_line_rc44data(float R, float G, float B, float siz 820
    ){if(siz==0){size:siz;} 820
    data[count]="#function name: Bezier_surface_control_line_rc44 4 sources"; 820
    count++;data[count]="#R: "+Float.toString(R); count++;data[count]="# 820
        G: "+Float.toString(G); count++; 820
        data[count]="#B: "+Float.toString(B); count++;data[count]="#siz: "+Float. 820
            toString(siz); count++; 820
            for (int i=0;i<4;i++) {for(int j=1;j<4;j++) {line3Ddata(S[i][j-1][0],S[i] 820
                [j-1][1],S[i][j-1][2],S[i][j][0],S[i][j][1],S[i][j][2],siz,R,G,B,1) 820
                ;}} 820
                for (int j=0;j<4;j++) {for(int i=1;i<4;i++) {line3Ddata(S[i-1][j][0],S[i] 820
                    [-1][j][1],S[i-1][j][2],S[i][j][0],S[i][j][1],S[i][j][2],siz,R,G,B,1) 820
                    ;}}} 820
void Bezier_surface_control_line_rc33data(float R, float G, float B, float siz 820
    ){if(siz==0){size:siz;} 820
    data[count]="#function name: Bezier_surface_control_line_rc33 4 sources"; 820
    count++;data[count]="#R: "+Float.toString(R); count++;data[count]="# 820
        G: "+Float.toString(G); count++; 820
        data[count]="#B: "+Float.toString(B); count++;data[count]="#siz: "+Float. 820
            toString(siz); count++;
```

```

821 for(int i=0;i<3;i++) {for(int j=1;j<3;j++) {line3Ddata(S[i][j-1][0],S[871
822   ][j-1][1],S[i][j-1][2],S[i][j][0],S[i][j][1],S[i][j][2],siz,R,G,B,1972
823   };}
824 for(int j=0;j<3;j++) {for(int i=1;i<3;i++) {line3Ddata(S[i-1][j][0],S[i-1][j][1],S[i-1][j][2],S[i][j][0],S[i][j][1],S[i][j][2],siz,R,G,B,1973
825   };}
826 void Bezier_surface_line_rc44data(float R,float G,float B,float siz,float
827   gap){if(siz==0){siz=siz*size;}}
828 data[count]="#function name: Bezier_surface_line_rc44 5 sources"; count+
829 ++;data[count]="#R: "+Float.toString(R); count++;data[count]="#G: "+
830 Float.toString(G); count++;
831 data[count]="#B: "+Float.toString(B); count++;data[count]="#siz: "+Float.
832   toString(siz);count++;data[count]="#gap: "+Float.toString(gap);count+
833 ++;
834 int i,j;float u,v,max=0;float[] [] [] save=new float[1005][1005][3];
835 for(u=0;u<1;u+=gap){for(v=0;v<1;v+=gap){float sumx=0,sumy=0,sumz=0;
836   for(i=0;i<4;i++) {for(j=0;j<4;j++) {sumx+=com[3][i]*com[3][j]*power(u,i
837   )*power(v,j)*power(1-u,3-i)*power(v-1,v-3-j)*S[i][j][0];}
838   for(i=0;i<4;i++) {for(j=0;j<4;j++) {sumy+=com[3][i]*com[3][j]*power(u,i
839   )*power(v,j)*power(1-u,3-i)*power(v-1,v-3-j)*S[i][j][1];}
840   for(i=0;i<4;i++) {for(j=0;j<4;j++) {sumz+=com[3][i]*com[3][j]*power(u,i
841   )*power(v,j)*power(1-u,3-i)*power(v-1,v-3-j)*S[i][j][2];}
842   save[int(u*1000)][int(v*1000)][0]=sumx;save[int(u*1000)][int(v*1000)][1]=sumy;
843   save[int(u*1000)][int(v*1000)][2]=sumz;
844   data[count]="#u,v vertex: "+Integer.toString((int)(u*1000))+ " "+ Integer.
845   toString((int)(v*1000))+ " "+Float.toString(sumx)+" "+Float.
846   toString(sumy)+" "+Float.toString(sumz);count++;
847   if(max*max*sumx*sumy*sumy*sumz*sumz)/max=sqrt(sumx*sumx+sumy*sumy+
848   sumy*sumz*sumz);}}
849 data[count]="#maxdistance: "+Float.toString(max);count++;
850 for(u=0;u<1;u+=gap){float savx=0,savy=0,savz=0;
851   for(v=0;v<1;v+=gap){float k,sumx=save[int(u*1000)][int(v*1000)
852   ][0],sumy=save[int(u*1000)][int(v*1000)][1],sumz=save[int(u*1000)
853   ][int(v*1000)][2];
854   k=sqrt(sumx*sumx+sumy*sumy+sumz*sumz)/max;
855   point3D_RGBdata(sumx,sumy,sumz,(int)(R*sqrt(sqrt(k)*k))+50,(int)(G*
856   sqrt(sqrt(k)*k))+50,(int)(B*sqrt(sqrt(k)*k))+50,5,siz);Bpoint3D(sumx
857   ,sumy,sumz);
858   if(v==0){savx=sumx;savy=sumy;savz=sumz;sumx=0;sumy=0;sumz=0;}
859   else{line3Ddata(savx,savy,savz,sumx,sumy,sumz,siz,(int)(R*sqrt(sqrt(k
860   )*k))+50,(int)(G*sqrt(sqrt(k)*k))+50,(int)(B*sqrt(sqrt(k)*k))+50,3);
861   Bline3D(savx,savy,savz,sumx,sumy,sumz);savx=sumx;savy=sumy;savz=
862   sumz;sumx=0;sumy=0;sumz=0;}}
863 void Bezier_surface_line_rc33data(float R,float G,float B,float siz,float
864   gap){if(siz==0){siz=siz*size;}}
865 data[count]="#function name: Bezier_surface_line_rc33 5 sources"; count+
866 ++;data[count]="#R: "+Float.toString(R); count++;data[count]="#G: "+
867 Float.toString(G); count++;
868 data[count]="#B: "+Float.toString(B); count++;data[count]="#siz: "+Float.
869   toString(siz);count++;data[count]="#gap: "+Float.toString(gap);count+
870 ++;
871 int i,j;float u,v,max=0;float[] [] [] save=new float[1005][1005][3];
872 for(u=0;u<1.001;u+=gap){for(v=0;v<1.001;v+=gap){float sumx=0,sumy=0,
873   sumz=0;
874   for(i=0;i<3;i++) {for(j=0;j<3;j++) {sumx+=com[2][i]*com[2][j]*power(u,1001
875   )*power(v,j)*power(1-u-2,-j)*power(v-1,-v-2)*S[i][j][0];}
876   for(i=0;i<3;i++) {for(j=0;j<3;j++) {sumy+=com[2][i]*com[2][j]*power(u,1002
877   )*power(v,j)*power(1-u-2,-j)*power(v-1,-v-2)*S[i][j][1];}
878   for(i=0;i<3;i++) {for(j=0;j<3;j++) {sumz+=com[2][i]*com[2][j]*power(u,1003
879   )*power(v,j)*power(1-u-2,-j)*power(v-1,-v-2)*S[i][j][2];}
880   data[count]="#u,v vertex: "+Integer.toString((int)(u*1000))+ " "+ Integer.
881   toString((int)(v*1000))+ " "+Float.toString(sumx)+" "+Float.
882   toString(sumy)+" "+Float.toString(sumz);count++;
883   if(max*max*sumx*sumy*sumy*sumz*sumz)/max=sqrt(sumx*sumx+sumy*sumy+
884   sumy*sumz*sumz);}}
885 data[count]="#maxdistance: "+Float.toString(max);count++;
886 for(u=0;u<1.001;u+=gap){float savx=0,savy=0,savz=0;
887   for(v=0;v<1.001;v+=gap){float k,sumx=save[int(u*1000)][int(v*1000)
888   ][0],sumy=save[int(u*1000)][int(v*1000)][1],sumz=save[int(u*1000)
889   ][int(v*1000)][2];
890   k=sqrt(sumx*sumx+sumy*sumy+sumz*sumz)/max;
891   if(v==0){savx=sumx;savy=sumy;savz=sumz;sumx=0;sumy=0;sumz=0;}
892   else{line3Ddata(savx,savy,savz,sumx,sumy,sumz,siz,(int)(R*sqrt(sqrt(k
893   )*k))+50,(int)(G*sqrt(sqrt(k)*k))+50,(int)(B*sqrt(sqrt(k)*k))+50,3);
894   Bline3D(savx,savy,savz,sumx,sumy,sumz);savx=sumx;savy=sumy;savz=
895   sumz;sumx=0;sumy=0;sumz=0;}}
896 void Bezier_surface_line_rc33data(float R,float G,float B,float siz,float
897   gap){if(siz==0){siz=siz*size;}}
898 data[count]="#function name: Bezier_surface_line_rc33 5 sources"; count+
899 ++;data[count]="#R: "+Float.toString(R); count++;data[count]="#G: "+
900 Float.toString(G); count++;
901 data[count]="#B: "+Float.toString(B); count++;data[count]="#siz: "+Float.
902   toString(siz);count++;data[count]="#gap: "+Float.toString(gap);count+
903 ++;
904 int i,j;float u,v,max=0;float[] [] [] save=new float[1005][1005][3];
905 for(u=0;u<1.001;u+=gap){for(v=0;v<1.001;v+=gap){float sumx=0,sumy=0,
906   sumz=0;
907   for(i=0;i<3;i++) {for(j=0;j<3;j++) {sumx+=com[2][i]*com[2][j]*power(u,1001
908   )*power(v,j)*power(1-u-2,-j)*power(v-1,-v-2)*S[i][j][0];
909   for(i=0;i<3;i++) {for(j=0;j<3;j++) {sumy+=com[2][i]*com[2][j]*power(u,1002
910   )*power(v,j)*power(1-u-2,-j)*power(v-1,-v-2)*S[i][j][1];
911   for(i=0;i<3;i++) {for(j=0;j<3;j++) {sumz+=com[2][i]*com[2][j]*power(u,1003
912   )*power(v,j)*power(1-u-2,-j)*power(v-1,-v-2)*S[i][j][2];
913   data[count]="#u,v vertex: "+Integer.toString((int)(u*1000))+ " "+ Integer.
914   toString((int)(v*1000))+ " "+Float.toString(sumx)+" "+Float.
915   toString(sumy)+" "+Float.toString(sumz);count++;
916   if(max*max*sumx*sumy*sumy*sumz*sumz)/max=sqrt(sumx*sumx+sumy*sumy+
917   sumy*sumz*sumz);}}
918 data[count]="#maxdistance: "+Float.toString(max);count++;
919 for(u=0;u<1.001;u+=gap){float k,sumx=save[int(u*1000)][int(v*1000)
920   ][0],sumy=save[int(u*1000)][int(v*1000)][1],sumz=save[int(u*1000)
921   ][int(v*1000)][2];
922   k=sqrt(sumx*sumx+sumy*sumy+sumz*sumz)/max;
923   point3D_RGBdata(sumx,sumy,sumz,(int)(R*sqrt(sqrt(k)*k))+50,(int)(G*
924   sqrt(sqrt(k)*k))+50,(int)(B*sqrt(sqrt(k)*k))+50,5,siz);Bpoint3D(sumx
925   ,sumy,sumz);
926   if(v==0){savx=sumx;savy=sumy;savz=sumz;sumx=0;sumy=0;sumz=0;}
927   else{line3Ddata(savx,savy,savz,sumx,sumy,sumz,siz,(int)(R*sqrt(sqrt(k
928   )*k))+50,(int)(G*sqrt(sqrt(k)*k))+50,(int)(B*sqrt(sqrt(k)*k))+50,3);
929   Bline3D(savx,savy,savz,sumx,sumy,sumz);savx=sumx;savy=sumy;savz=
930   sumz;sumx=0;sumy=0;sumz=0;}}
931 void Bezier_triangle_control_line44data(float R,float G,float B,float siz{
932   int i,j,k;if(siz==0){siz=size;}}
933 data[count]="#function name: Bezier_triangle_control_line44 4 sources";
934 count++;data[count]="#R: "+Float.toString(R); count++;data[count]="#G: "+
935   Float.toString(G); count++;
936 data[count]="#B: "+Float.toString(B); count++;data[count]="#siz: "+Float.
937   toString(siz);count++;
938 for(i=0;i<3;i++) {for(j=0;j<3;j++) {if(i+j<2){k=3-i-j;line3Ddata(T[i][j
939   ][0],T[i][j][k][1],T[i][j][k][2],T[i][j+1][k-1][0],T[i][j+1][k
940   ][1],T[i][j+1][k][1],T[i][j+1][k][2],siz,R,G,B,1);}
941 for(j=0;j<3;j++) {for(k=0;k<3;k++) {if(k+j<2){i=3-k-j;line3Ddata(T[i][j
942   ][0],T[i][j][k][1],T[i][j][k][2],T[i+1][j][k+1][0],T[i+1][j][k+1][1]
943   [2],T[i+1][j][k+1][2],siz,R,G,B,1);}
944 for(k=0;k<3;k++) {for(i=0;i<3;i++) {if(i+k<2){j=3-k-i;line3Ddata(T[i][j
945   ][0],T[i][j][k][1],T[i][j][k][2],T[i+1][j][k+1][0],T[i+1][j][k+1][1]
946   [2],T[i+1][j][k+1][2],siz,R,G,B,1);}}}
947 void Bezier_triangle_line44data(float R,float G,float B,float siz{
948   int i,j,k;if(siz==0){siz=size;}}
949 data[count]="#function name: Bezier_triangle_line44 5 sources"; count++;
950 data[count]="#R: "+Float.toString(R); count++;data[count]="#G: "+
951   Float.toString(G); count++;
952 data[count]="#B: "+Float.toString(B); count++;data[count]="#siz: "+Float.
953   toString(siz);count++;data[count]="#gap: "+Float.toString(gap);count+
954 ++;
955 int i,j,k;float s,t,u,max=0;float[] [] [] save=new float
956   [1005][1005][3];
957 for(s=0;s<1;s+=gap){for(t=0;t<1;t+=gap){if(s>t){continue;}u=1-s-t;
958   float sumx=0,sumy=0,sumz=0;
959   for(i=0;i<3;i++) {for(j=0;j<3;j++) {if(j=0;j<3;j+){if(i+j<3){k=3-i-j;sumx+=com[2][i][j][k][0];}}
960   for(i=0;i<3;i++) {for(j=0;j<3;j++) {if(j=0;j<3;j+){if(i+j<3){k=3-i-j;sumy+=com[2][i][j][k][1];}}
961   for(i=0;i<3;i++) {for(j=0;j<3;j++) {if(j=0;j<3;j+){if(i+j<3){k=3-i-j;sumz+=com[2][i][j][k][2];}}
962   save[int(s*1000)][int(t*1000)][0]=sumx;save[int(s*1000)][int(t*1000)][1]=sumy;
963   save[int(s*1000)][int(t*1000)][2]=sumz;
964   data[count]="#s,t,u vertex: "+Integer.toString((int)(s*1000))+ " "+ Integer.
965   toString((int)(t*1000))+ " "+Integer.toString((int)(u*1000))+ " "+Float.toString(
966   sumx)+" "+Float.toString(sumy)+" "+Float.toString(sumz);count++;
967   if(max*max*sumx*sumy*sumy*sumz*sumz)/max=sqrt(sumx*sumx+sumy*sumy+
968   sumy*sumz*sumz);}}
969 data[count]="#maxdistance: "+Float.toString(max);count++;
970 for(s=0;s<1;s+=gap){float gap=1.0f;for(t=0;t<1;t+=gap){if(s>t){con
971   tinue;}u=1-s-t;
972   float k1,sumx=save[int(s*1000)][int(t*1000)][int(u*1000)][0],
973   sumy=save[int(s*1000)][int(t*1000)][int(u*1000)][1],sumz=save[int(s*1000)
974   ][int(t*1000)][int(u*1000)][2];
975   k1=sqrt(sumx*sumx+sumy*sumy+sumz*sumz)/max;
976   point3D_RGBdata(sumx,sumy,sumz,(int)(R*sqrt(sqrt(k1)*k1))+50,(int)(G*
977   sqrt(sqrt(k1)*k1))+50,(int)(B*sqrt(sqrt(k1)*k1))+50,5,siz);Bpoint3D(sumx
978   ,sumy,sumz);
979   if(v==0){savx=sumx;savy=sumy;savz=sumz;sumx=0;sumy=0;sumz=0;}
980   else{line3Ddata(savx,savy,savz,sumx,sumy,sumz,siz,(int)(R*sqrt(sqrt(k1
981   )*k1))+50,(int)(G*sqrt(sqrt(k1)*k1))+50,(int)(B*sqrt(sqrt(k1)*k1))+50,3);
982   Bline3D(savx,savy,savz,sumx,sumy,sumz);savx=sumx;savy=sumy;savz=
983   sumz;sumx=0;sumy=0;sumz=0;}}
984 void Bezier_triangle_control_line44data(float R,float G,float B,float siz{
985   int i,j,k;if(siz==0){siz=size;}}
986 data[count]="#function name: Bezier_triangle_control_line44 4 sources";
987 count++;data[count]="#R: "+Float.toString(R); count++;data[count]="#G: "+
988   Float.toString(G); count++;
989 data[count]="#B: "+Float.toString(B); count++;data[count]="#siz: "+Float.
990   toString(siz);count++;data[count]="#gap: "+Float.toString(gap);count+
991 ++;
```

```

+>1){continue;};u+=1-s;t-
float k1,sumx+=save[[int](s+100)][[int](t+100)][[int](u+100)][0],sumy+=save[[int](s+100)][[int](t+100)][[int](u+100)][1],sumz+=save[[ 959
int](s+100)][[int](t+100)][[int](u+100)][2]; 960
k1+=sqrt((sumx*sumx+sumy*sumy+sumz*sumz)/max); 961
point3D_RGBdata(sumx,sumy,sumz,(int)(R*sqrt(sqrt(k1*k1))+50,(int)( 962
G*sqrt(sqrt(k1*k1))+50,(int)(B*sqrt(sqrt(k1*k1))+50,5,siz); 963
Bpoint3D(sumx,sumy,sumz); 964
if(s==0){sumx+=sumy;sumy+=sumz;sumx+=sumy;sumz=0;sumy=0;sumz=0;} 965
else{line3Ddata(savx,savy,savz,sumx,sumy,sumz,siz,(int)(R*sqrt(sqrt( 966
k1*k1))+50,(int)(G*sqrt(sqrt(k1*k1))+50,(int)(B*sqrt(sqrt(k1*k1)+963
+50,3); 967
Bline3D(savx,savy,savz,sumx,sumy,sumz);savx+=sumx;savy+=sumy;savz= 968
sumx;sumy=0;sumz=0;}}; 969
for(u=0;u<1;u+=gap){float savx0,savy0,savz0;for(t=0;t<1;t+=gap){if(t< 970
+>1){continue;s+=1-u;t-;
float k1,sumx+=save[[int](s+100)][[int](t+100)][[int](u+100)][0], 971
sumy+=save[[int](s+100)][[int](t+100)][[int](u+100)][1],sumz+=save[[ 972
int](s+100)][[int](t+100)][[int](u+100)][2]; 973
k1+=sqrt((sumx*sumx+sumy*sumy+sumz*sumz)/max); 974
point3D_RGBdata(sumx,sumy,sumz,(int)(R*sqrt(sqrt(k1*k1))+50,(int)( 975
G*sqrt(sqrt(k1*k1))+50,(int)(B*sqrt(sqrt(k1*k1))+50,5,siz); 976
Bpoint3D(sumx,sumy,sumz);
if(t==0){sumx+=sumy;sumy+=sumz;sumx+=sumy;sumz=0;sumy=0;sumz=0;} 977
else{line3Ddata(savx,savy,savz,sumx,sumy,sumz,siz,(int)(R*sqrt(sqrt( 978
k1*k1))+50,(int)(G*sqrt(sqrt(k1*k1))+50,(int)(B*sqrt(sqrt(k1*k1)+975
+50,3); 979
Bline3D(savx,savy,savz,sumx,sumy,sumz);savx+=sumx;savy+=sumy;savz= 980
sumx;sumy=0;sumz=0;}}; 981
for(s=0;s<1;s+=gap){float savx0,savy0,savz0;for(u=0;u<1;u+=gap){if(s< 982
+>1){continue;s+=1-u;t-;
float k1,sumx+=save[[int](s+100)][[int](t+100)][[int](u+100)][0], 983
sumy+=save[[int](s+100)][[int](t+100)][[int](u+100)][1],sumz+=save[[ 984
int](s+100)][[int](t+100)][[int](u+100)][2];
k1+=sqrt((sumx*sumx+sumy*sumy+sumz*sumz)/max); 985
point3D_RGBdata(sumx,sumy,sumz,(int)(R*sqrt(sqrt(k1*k1))+50,(int)( 986
G*sqrt(sqrt(k1*k1))+50,(int)(B*sqrt(sqrt(k1*k1))+50,5,siz); 987
Bpoint3D(sumx,sumy,sumz);
if(u==0){sumx+=sumy;sumy+=sumz;sumx+=sumy;sumz=0;sumy=0;sumz=0;} 988
else{line3Ddata(savx,savy,savz,sumx,sumy,sumz,siz,(int)(R*sqrt(sqrt( 989
k1*k1))+50,(int)(G*sqrt(sqrt(k1*k1))+50,(int)(B*sqrt(sqrt(k1*k1)+986
+50,3); 990
Bline3D(savx,savy,savz,sumx,sumy,sumz);savx+=sumx;savy+=sumy;savz= 991
sumx;sumy=0;sumz=0;}}; 992
void Bezier_triangle_control_line33data(float R,float G,float B,float siz){ 993
int i,j,k;if(siz==0){siz=size;}
data[count]="#function name: Bezier_triangle_control_line33 4 sources"; 994
count++;data[count]="#R:#"+Float.ToString(R);count++;data[count]="# 995
G:#"+Float.ToString(G);count++; 996
data[count]="#B:#"+Float.ToString(B);count++;data[count]="#siz: "+Float 997
.ToString(siz);count++;
for(i=0;i<2;i++){for(j=0;j<2;j++){if(i+j<1){k=2-i-j;line3Ddata(T[i][j] 998
[1][0],T[i][j][k][1],T[i][j][k][2],T[i][j+1][k-1][0],T[i][j+1][k- 999
-1][1],T[i][j+1][k-1][k-2][2],siz,R,G,B,1,)}}
for(j=0;j<2;j++){for(k=0;k<2;k++){if(k+j<1){i=2-k-j;line3Ddata(T[i][j] 1000
[1][0],T[i][j][k][1],T[i][j][k][2],T[i+1][j][k+1][0],T[i+1][j][k- 1001
+1][1],T[i+1][j][k-1][k][2],siz,R,G,B,1,)}}
for(k=0;k<2;k++){for(i=0;i<2;i++){if(i+k<1){j=2-i-k;line3Ddata(T[i][j] 1002
[1][0],T[i][j][k][1],T[i][j][k][2],T[i+1][j-1][k][0],T[i+1][j-1][k- 1003
+1],T[i+1][j-1][k][2],siz,R,G,B,1,)}}
void Bezier_triangle_line33data(float R,float G,float B,float siz,float gap 1004
){if(siz==0){siz=size;}
data[count]="#function name: Bezier_triangle_line33 5 sources",count++; 1005
data[count]="#R:#"+Float.ToString(R);count++;data[count]="#G:#"+ 1006
Float.ToString(G);count++; 1007
data[count]="#B:#"+Float.ToString(B);count++;data[count]="#siz: "+Float 1008
.ToString(siz);count++;data[count]="#gap: "+Float.ToString(gap);count++; 1009
int i,j,k;float s,t,u,max=0;float [][] [] [] save2=new float [1010][105][105][3];
for(s=0;s<1;s+=gap){for(t=0;t<1;t+=gap){if(s+t>1){continue;};u=1.0-s-t; 1011
float sumx=0,sumy=0,sumz=0;
for(i=0;i<2;i++){for(j=0;j<2;j++){if(i+j<2){k=2-i-j;sumx+=com2[i 1012
[1][j][k]*power(s,i)*power(t,j)*power(u,k)*T[i][j][k][0];}}
for(i=0;i<2;i++){for(j=0;j<2;j++){if(i+j<2){k=2-i-j;sumy+=com2[i 1013
[1][j][k]*power(s,i)*power(t,j)*power(u,k)*T[i][j][k][1];}}
for(i=0;i<2;i++){for(j=0;j<2;j++){if(i+j<2){k=2-i-j;sumz+=com2[i 1014
[1][j][k]*power(s,i)*power(t,j)*power(u,k)*T[i][j][k][2];}}
save2[[int](s+10)*[int](t+10)*[int](u+10)*[int](0)=sumx;save2[[int](s+10)*[int](t+10)*[int](u+10)*[int](1)=sumy;save2[[int](s+10)*[int](t+10)*[int](u+10)*[int](2)=sumz;
data[count]="#s_t,u vertex: "+Integer.ToString((int)(s+10)*10)+" 1015
Integer.ToString((int)(t+10)*10)+"+Integer.ToString((int)(u+10)*10)+" 1016
*10)+" "+Float.ToString(Sumx)+" "+Float.ToString(Sumy)+" "+Float 1017
.ToString(Sumz);count++;
if(max>=sumx+sumy+sumz){max=sqrt((sumx*sumx+sumy* 1018
sumy+sumz*sumz));}
data[count]="#maxdistance: "+Float.ToString(max);count++; 1019
for(s=0;s<1;s+=gap){float savx0,savy0,savz0;for(t=0;t<1;t+=gap){if(s 1020
+>1){continue;s+=1-u;1.0-s-t;
float k1,sumx+=save2[[int](s+10)*10][[int](t+10)*10][[int](u+10) 1021
*[10][0],sumy+=save2[[int](s+10)*10][[int](t+10)*10][[int](u+10) 1022
*[10][1],sumz+=save2[[int](s+10)*10][[int](t+10)*10][[int](u+10) 1023
*[10][2];
k1+=sqrt((sumx*sumx+sumy*sumy+sumz*sumz)/max);point3D_RGBdata(sumx, 1024
sumy,sumz,(int)(R*sqrt(sqrt(k1*k1))+50,(int)(G*sqrt(sqrt(k1*k1))+ 1025
+50,(int)(B*sqrt(sqrt(k1*k1))+50,5,siz);Bpoint3D(sumx,sumy,sumz); 1026
if(t==0){savx+=sumx;savy+=sumy;savz+=sumz;sumx=0;sumy=0;sumz=0;}}; 1027
else{line3Ddata(savx,savy,savz,sumx,sumy,sumz,siz,(int)(R*sqrt(sqrt( 1028
k1*k1))+50,(int)(G*sqrt(sqrt(k1*k1))+50,(int)(B*sqrt(sqrt(k1*k1)+ 1025
+50,3);Bline3D(savx,savy,savz,sumx,sumy,sumz);savx+=sumx;savy+=sumy;savz= 1029
sumx;sumy=0;sumz=0;}}; 1030
for(t=0;t<1;t+=gap){float savx,savy,savz;for(u=0;u<1;u+=gap){if(u< 1031
+>1){continue;s+=1-u;1.0-u;
float k1,sumx+=save2[[int](s+10)*10][[int](t+10)*10][[int](u+10) 1032
*[10][0],sumy+=save2[[int](s+10)*10][[int](t+10)*10][[int](u+10) 1033
*[10][1],sumz+=save2[[int](s+10)*10][[int](t+10)*10][[int](u+10) 1034
*[10][2];
k1+=sqrt((sumx*sumx+sumy*sumy+sumz*sumz)/max);point3D_RGBdata(sumx, 1035
sumy,sumz,(int)(R*sqrt(sqrt(k1*k1))+50,(int)(G*sqrt(sqrt(k1*k1))+ 1036
+50,(int)(B*sqrt(sqrt(k1*k1))+50,5,siz);Bpoint3D(sumx,sumy,sumz); 1037
if(u==0){savx+=sumx;savy+=sumy;savz+=sumz;sumx=0;sumy=0;sumz=0;}}; 1038
else{line3Ddata(savx,savy,savz,sumx,sumy,sumz,siz,(int)(R*sqrt(sqrt( 1039
k1*k1))+50,(int)(G*sqrt(sqrt(k1*k1))+50,(int)(B*sqrt(sqrt(k1*k1)+ 1036
+50,3);Bline3D(savx,savy,savz,sumx,sumy,sumz);savx+=sumx;savy+=sumy;savz= 1040
sumx;sumy=0;sumz=0;}}; 1041
for(u=0;u<1;u+=gap){float savx0,savy0,savz0;for(t=0;t<1;t+=gap){if(t< 1042
+>1){continue;s+=1-u-t;
float k1,sumx+=save2[[int](s+10)*10][[int](t+10)*10][[int](u+10) 1043
*[10][0],sumy+=save2[[int](s+10)*10][[int](t+10)*10][[int](u+10) 1044
*[10][1],sumz+=save2[[int](s+10)*10][[int](t+10)*10][[int](u+10) 1045
*[10][2];
k1+=sqrt((sumx*sumx+sumy*sumy+sumz*sumz)/max);point3D_RGBdata(sumx, 1046
sumy,sumz,(int)(R*sqrt(sqrt(k1*k1))+50,(int)(G*sqrt(sqrt(k1*k1))+ 1047
+50,(int)(B*sqrt(sqrt(k1*k1))+50,5,siz);Bpoint3D(sumx,sumy,sumz); 1048
if(t==0){savx+=sumx;savy+=sumy;savz+=sumz;sumx=0;sumy=0;sumz=0;}}; 1049
else{line3Ddata(savx,savy,savz,sumx,sumy,sumz,siz,(int)(R*sqrt(sqrt( 1050
k1*k1))+50,(int)(G*sqrt(sqrt(k1*k1))+50,(int)(B*sqrt(sqrt(k1*k1)+ 1047
+50,3);Bline3D(savx,savy,savz,sumx,sumy,sumz);savx+=sumx;savy+=sumy;savz= 1051
sumx;sumy=0;sumz=0;}}; 1052
void example_Bezier_rectdata(){
data[count]="#function name: example_Bezier_rectdata 0 sources";count++; 1053
checkbox33data(0,-200,-200,-100,-300,0,-200,-200,200); 1054
checkbox33data(1,100,-100,-200,-200,-200,-100,-100,200); 1055
checkbox33data(2,200,200,-200,-300,100,0,200,200,200); 1056
Bezier_surface_line_rc33data(255,0,size,0.2); 1057
Bezier_surface_control_line_rc33data(237,0,255,size); 1058
void example_Bezier_tridata(){
data[count]="#function name: example_Bezier_rectdata 0 sources";count++; 1059
checkbox33_sampledata();
Bezier_triangle_line33data(255,0,size,0.2); 1060
Bezier_triangle_control_line33data(237,0,255,size); 1061

```

```

1004 void example_Bezier_linedata(){data[count]="#function name:  
example_Bezier_linedata 0 sources";count++;Bezier3Ddata 1064  
(100,100,100,-100,-100,50,200,-150,1,0); 1065  
1005 void example_Bezier_rect44data(){ 1066  
data[count]="#function name: example_Bezier_linedata 0 sources";count++#068 1067  
checkrc44data(0,123,231,321,241,-123,57,-13,193,200,-147,-100,0); 1068  
checkrc44data(1,152,-179,-35,294,240,-199,-201,144,-44,208,-98,273)969  
checkrc44data(2,71,-163,258,111,-222,260,109,-10,63,99,-130,91); 1070  
checkrc44data(3,-100,210,57,-39,-284,95,-68,35,249,-103,-47,38); 1070  
1009 Bezier_surface_line_rc44data(255,0,0,size_0.05); 1071  
Bezier_surface_control_line_rc44data(255,246,142,size); 1072  
1010 void example_Bezier_rect33data(){ 1073  
data[count]="#function name: example_Bezier_rect33data 0 sources";count 1074  
++; 1075  
checkrc33data(0,21,241,-123,57,-13,193,200,-147,-100);checkrc33data  
(1,-35,294,240,-199,-201,144,-44,208,-96);checkrc33data 1076  
(2,258,111,-222,260,109,-10,63,99,-130); 1077  
Bezier_surface_line_rc33data(255,0,0,size_0.05); 1078  
Bezier_surface_control_line_rc33data(255,246,142,size); 1078  
1015 void example_Bezier_tri44data(){ 1079  
data[count]="#function name: example_Bezier_tri44data 0 sources";count++ 1080  
checkrc44data(); 1080  
Bezier_triangle_line44data(255,0,0,size_0.1); 1081  
Bezier_triangle_control_line44data(255,246,142,size); 1082  
1019 void example_Bezier_tris33data(){ data[count]="#function name:  
example_Bezier_tris33data 0 sources";count++;checkrc33data(); 1083  
Bezier_triangle_line33data(0,255,0,0,size_0.2); 1084  
Bezier_triangle_control_line33data(255,246,142,size); 1085  
1022 void checkrc44data(int m,float x1,float y1,float z1,float x2,float y2,float 1086  
z2,float x3,float y3,float z3,float x4,float y4,float z4){ 1087  
data[count]="#function name: checkrc44data 13 sources";count++; 1088  
data[count]="#": "#Integer.toString(m); count++; 1088  
data[count]="#x1": "#Float.toString(x1); count++;data[count]="#y1": "#Float 1089  
.toString(y1); count++;data[count]="#z1": "#Float.toString(z1); count 1089  
++; 1090  
data[count]="#x2": "#Float.toString(x2); count++;data[count]="#y2": "#Float 1090  
.toString(y2); count++;data[count]="#z2": "#Float.toString(z2); count 1091  
++; 1091  
data[count]="#x3": "#Float.toString(x3); count++;data[count]="#y3": "#Float 1092  
.toString(y3); count++;data[count]="#z3": "#Float.toString(z3); count 1093  
++; 1093  
data[count]="#x4": "#Float.toString(x4); count++;data[count]="#y4": "#Float 1094  
.toString(y4); count++;data[count]="#z4": "#Float.toString(z4); count 1095  
++; 1095  
S[m][0][0]=x1;S[m][0][1]=y1;S[m][0][2]=z1; 1096  
S[m][1][0]=x2;S[m][1][1]=y2;S[m][1][2]=z2; 1096  
S[m][2][0]=x3;S[m][2][1]=y3;S[m][2][2]=z3; 1096  
S[m][3][0]=x4;S[m][3][1]=y4;S[m][3][2]=z4; 1096  
1097 void checkcrc33data(int m,float x,y1,float y1,float z1,float x2,float y2,float  
z2,float x3,float y3,float z3){ 1098  
data[count]="#function name: checkcrc33 13 sources";count++; 1098  
data[count]="#": "#Integer.toString(m); count++; 1099  
data[count]="#x1": "#Float.toString(x1); count++;data[count]="#y1": "#Float 1100  
.toString(y1); count++;data[count]="#z1": "#Float.toString(z1); count 1100  
++; 1101  
data[count]="#x2": "#Float.toString(x2); count++;data[count]="#y2": "#Float 1102  
.toString(y2); count++;data[count]="#z2": "#Float.toString(z2); count 1102  
++; 1102  
data[count]="#x3": "#Float.toString(x3); count++;data[count]="#y3": "#Float 1103  
.toString(y3); count++;data[count]="#z3": "#Float.toString(z3); count 1103  
++; 1103  
S[m][0][0]=x1;S[m][0][1]=y1;S[m][0][2]=z1; 1103  
S[m][1][0]=x2;S[m][1][1]=y2;S[m][1][2]=z2; 1103  
S[m][2][0]=x3;S[m][2][1]=y3;S[m][2][2]=z3; 1103  
1044 void checktdata(int i,int j,int k,float x,float y,float z){ 1104  
data[count]="#function name: check 6 sources";count++; 1105  
data[count]="#": "#Integer.toString(i); count++;data[count]="#j": "#  
Integer.toString(j); count++;data[count]="#k": "#Integer.toString(k);106  
count++; 1107  
data[count]="#": "#Float.toString(x); count++;data[count]="#": "#Float.  
toString(y); count++;data[count]="#": "#Float.toString(z); count++; 108  
T[i][j][k][0]=x;T[i][j][k][1]=y;T[i][j][k][2]=z; 1109  
1110  
1051 void check3Ddata(float x,float y,float z,float siz){if(siz==0){siz=size;} 1110  
data[count]="#function name: check3D 4 sources";count++; 1111  
data[count]="#": "#Float.toString(x); count++;data[count]="#": "#Float.  
toString(y); count++;data[count]="#": "#Float.toString(z); count++; 1112  
data[count]="#siz": "#Float.toString(size); count++; 1112  
A[0]=-x*cos(radians(ceta));A[0]=A[0]*siz; 1113  
A[1]=-y*cos(radians(pi))-x*sin(radians(pi))*sin(radians(ceta))-z*cos( 1113  
radians(ceta))*sin(radians(pi));A[1]=A[1]*siz; 1114  
1115  
void checktr44data(){ 1116  
data[count]="#function name: checktr44 0 sources";count++; 1117  
checktdata(3,0,0,100,100,100); 1118  
checktdata(2,1,0,50,75,125);checktdata 1119  
(2,0,1,75,-75,50); 1120  
checktdata(1,2,0,104,115,92);checktdata(1,1,1,4,-104,93); 1121  
checktdata(1,0,2,-231,-131,31); 1122  
checktdata(0,3,0,-100,0,-30);checktdata(0,2,1,-60,-30,-130);checktdata 1123  
(0,1,2,-30,130,-50);checktdata(0,0,3,104,-90,-100); 1123

```



```

1244 (size==0){size=size;};
1245 for(int i=0;i<4;j++) {for(int j=1;j<4;j++) {line3D(S[i][j-1][0],S[i][j][0],
1246 [j-1][1],S[i][j-1][2],S[i][j][0],S[i][j][1],S[i][j][2],size,R,G,B,1,);}
1247 for(int j=0;j<4;j++) {for(int i=1;i<4;i++) {line3D(S[i-1][j][0],S[i][j][1][0],
1248 [j][1],S[i-1][j][1],S[i][j][0],S[i][j][1],S[i][j][2],size,R,G,B,1,);}
1249 for(int j=0;j<3;j++) {for(int i=1;i<3;i++) {line3D(S[i-1][j][0],S[i][j][1][0],
1250 [j][1],S[i-1][j][1],S[i][j][0],S[i][j][1],S[i][j][2],size,R,G,B,1,);}}
1251 void Bezier_surface_control_line_r33(float R,float G,float B,float siz,float gap){ 1303
1252 if(size==0){size=size;}; 1304
1253 for(int i=0;i<1;i++) {for(int j=1;j<3;j++) {line3D(S[i][j-1][0],S[i][j][0],
1254 [j-1][1],S[i][j-1][2],S[i][j][0],S[i][j][1],S[i][j][2],size,R,G,B,1,);}
1255 for(int j=0;j<3;j++) {for(int i=1;i<4;i++) {line3D(S[i-1][j][0],S[i][j][1][0],
1256 [j][1],S[i-1][j][1],S[i][j][0],S[i][j][1],S[i][j][2],size,R,G,B,1,);}
1257 for(int i=0;i<3;j++) {for(int i=1;i<3;i++) {line3D(S[i-1][j][0],S[i][j][1][0],
1258 [j][1],S[i-1][j][1],S[i][j][0],S[i][j][1],S[i][j][2],size,R,G,B,1,);}}
1259 void Bezier_surface_line_rc44(float R,float G,float B,float siz,float gap){ 1306
1260 if(size==0){size=size;}; 1307
1261 //fill(255);TextSize(20);text("scale=sqrt(sqrt(k)*k)",413,70);
1262 int i,j,float u,v,max=0;float [] [] [] save=new float[1005][1005][3];
1263 for(u=0;u<1;u+gap){for(v=0;v<1;v+gap){float sum=0,sumy=0,sumz=0; 1308
1264 for(i=0;i<4;i++) {for(j=0;j<4;j++) {sum+=com[3][i]*com[3][j]*power(u,i*109
1265 )*power(v,j)*power(1-u,i-3)*power(1-v,j-3)*S[i][j][0];}
1266 for(i=0;i<4;i++) {for(j=0;j<4;j++) {sum+=com[3][i]*com[3][j]*power(u,i*110
1267 )*power(v,j)*power(1-u,i-3)*power(1-v,j-3)*S[i][j][1];}
1268 for(i=0;i<4;i++) {for(j=0;j<4;j++) {sum+=com[3][i]*com[3][j]*power(u,i*111
1269 )*power(v,j)*power(1-u,i-3)*power(1-v,j-3)*S[i][j][2];}}
1270 save[(int)(u*1000)][(int)(v*1000)][0]=sumx;save[(int)(u*1000)][(int)(v*1000)][1]=sumy;
1271 save[(int)(u*1000)][(int)(v*1000)][2]=sumz;
1272 if(max*sumx+sumy*sumy+sumy*sumz+sumz*sumz){max=sqrt(sumx*sumx+sumy* 1313
1273 sumy*sumz*sumz);}}
1274 for(u=0;u<1;u+gap){float savx=0,savy=0,savz=0; 1314
1275 for(v=0;v<1;v+gap){float k,savx=save[(int)(u*1000)][(int)(v*1000)][0],sumx=save[(int)(u*1000)][(int)(v*1000)][1],sumz=save[(int)(u*1000)][(int)(v*1000)][2]; 1315
1276 k=sqrt(sumx*sumx+sumy*sumy+sumz*sumz); 1316
1277 point3D_RGB(sumx,sumy,sumz,(int)(R*sqrt(sqrt(k)*k))+50,(int)(G*sqrt( 1317
1278 sqrt(k)*k))+50,(int)(B*sqrt(sqrt(k)*k))+50,5,siz);
1279 if(v==0){savx=0;savy=0;savz=0;sumx=0;sumy=0;sumz=0;} 1318
1280 else{line3D(savx,savx,savz,sumx,sumy,sumz,(int)(R*sqrt(sqrt(k)*k)*
1281 +50,(int)(G*sqrt(sqrt(k)*k))+50,(int)(B*sqrt(sqrt(k)*k))+50,3); 1319
1282 savx+=sumx;savx+=sumy;savz+=sumx;sumy=0;sumz=0;}}}
1283 for(u=0;u<1;u+gap){float k,savx=0,savy=0,savz=0; 1320
1284 for(v=0;v<1;v+gap){float k,savx=save[(int)(u*1000)][(int)(v*1000)][0],sumx=save[(int)(u*1000)][(int)(v*1000)][1],sumz=save[(int)(u*1000)][(int)(v*1000)][2]; 1321
1285 k=sqrt(sumx*sumx+sumy*sumy+sumz*sumz); 1322
1286 if(u==0){savx=0;savy=0;savz=0;sumx=0;sumy=0;sumz=0;} 1323
1287 else{line3D(savx,savx,savz,sumx,sumy,sumz,(int)(R*sqrt(sqrt(k)*k)*
1288 +50,(int)(G*sqrt(sqrt(k)*k))+50,(int)(B*sqrt(sqrt(k)*k))+50,3); 1324
1289 savx+=sumx;savx+=sumy;savz+=sumx;sumy=0;sumz=0;}}}
1290 void Bezier_surface_line_rc33(float R,float G,float B,float siz,float gap){ 1325
1291 if(size==0){size=size;}; 1326
1292 for(int i=0;u<1.001;u+gap){for(v=0;v<1.001;v+gap){float sum=0,sumy=0, 1326
1293 sumz=0; 1327
1294 for(i=0;i<1;i++) {for(j=0;j<3;j++) {sumx+=com[2][i]*com[2][j]*power(u,i
1295 )*power(v,j)*power(1-u,i-2)*power(1-v,j-2)*S[i][j][0];}
1296 for(i=0;i<3;i++) {for(j=0;j<3;j++) {sumx+=com[2][i]*com[2][j]*power(u,i*129
1297 )*power(v,j)*power(1-u,i-2)*power(1-v,j-2)*S[i][j][1];}
1298 for(i=0;i<3;i++) {for(j=0;j<3;j++) {sumx+=com[2][i]*com[2][j]*power(u,i*130
1299 )*power(v,j)*power(1-u,i-2)*power(1-v,j-2)*S[i][j][2];}}
1300 save[(int)(u*1000)][(int)(v*1000)][0]=sumx;save[(int)(u*1000)][(int)(v*1000)][1]=sumy;
1301 save[(int)(u*1000)][(int)(v*1000)][2]=sumz;
1302 for(u=0;u<1.001;u+gap){float savx=0,savy=0,savz=0; 1331
1303 for(v=0;v<1.001;v+gap){float sumx=save[(int)(u*1000)][(int)(v*1000)][0],sumy=save[(int)(u*1000)][(int)(v*1000)][1],sumz=save[(int)(u*1000)][(int)(v*1000)][2]; 1332
1304 point3D_RGB(sumx,sumy,sumz,R,G,B,5,siz);
1305 if(v==0){savx=0;savy=0;savz=0;sumx=0;sumy=0;sumz=0;} 1333
1306 else{line3D(savx,savx,savz,sumx,sumy,sumz,(int)(R*sqrt(sqrt(k)*k)*
1307 +50,(int)(G*sqrt(sqrt(k)*k))+50,(int)(B*sqrt(sqrt(k)*k))+50,3); 1334
1308 savx+=sumx;savx+=sumy;savz+=sumx;sumy=0;sumz=0;}}}
1309 for(u=0;u<1.001;u+gap){float k,savx=0,savy=0,savz=0; 1335
1310 for(v=0;v<1.001;v+gap){float k,savx=save[(int)(u*1000)][(int)(v*1000)][0],sumx=save[(int)(u*1000)][(int)(v*1000)][1],sumz=save[(int)(u*1000)][(int)(v*1000)][2]; 1336
1311 k=sqrt(sumx*sumx+sumy*sumy+sumz*sumz); 1337
1312 if(u==0){savx=0;savy=0;savz=0;sumx=0;sumy=0;sumz=0;} 1337
1313 else{line3D(savx,savx,savz,sumx,sumy,sumz,(int)(R*sqrt(sqrt(k)*k)*
1314 +50,(int)(G*sqrt(sqrt(k)*k))+50,(int)(B*sqrt(sqrt(k)*k))+50,3); 1338
1315 savx+=sumx;savx+=sumy;savz+=sumx;sumy=0;sumz=0;}}}
1316 void Bezier_triangle_control_line44(float R,float G,float B,float siz,int 1340
1317 i,j,k;if(size==0){size=size;}; 1340
1318 for(i=0;i<3;j++) {for(j=0;j<3;j++) {(if(i+j<2)=(k=3-i-1)*line3D(T[i][j][k
1319 [1][0],T[i][j][1][1],T[i][j][2][1],T[i][j][1][1]-k-1][0],T[i][j][1][k
1320 [-1][1],T[i][j][1][1]-k-1][2],size,R,G,B,1,));}
1321 for(j=0;j<3;j++) {for(k=0;k<3;k++) {(if(k+j<2)=(i=3-k-1)*line3D(T[i][j][k
1322 [1][0],T[i][j][k][1][1],T[i][j][k][2][1],T[i][j][k][1][0],T[i][j][k
1323 [-1][1],T[i][j][k][1][2],size,R,G,B,1,));}
1324 for(k=0;k<3;k++) {for(i=0;i<3;i++) {for(j=0;j<3;j++) {(if(i+j<2)=(j=3-i-1)*line3D(T[i][j][k
1325 [1][0],T[i][j][k][1][1],T[i][j][k][2][1],T[i][j][k][1][0],T[i][j][k
1326 [-1][1],T[i][j][k][1][2],size,R,G,B,1,));}
1327 for(i=0;i<3;i++) {for(j=0;j<3;j++) {(if(i+j<2)=(j=3-i-1)*line3D(T[i][j][k
1328 [1][0],T[i][j][k][1][1],T[i][j][k][2][1],T[i][j][k][1][0],T[i][j][k
1329 [-1][1],T[i][j][k][1][2],size,R,G,B,1,));}}
1330 void Bezier_triangle.line44(float R,float G,float B,float siz,float gap){ 1345
1331 if(size==0){size=size;}; 1346
1332 int i,j,k;float s,t,u,max=0;float [] [] [] save=new float[106][106][3];
1333 for(i=0;i<1;i++) {for(j=0;j<1;j++) {for(k=0;k<1;k++) {for(l=0;l<1;l++)

```

```

1347 void Bezier_triangle_line33(float R,float G,float B,float siz,float gap){ 1387
1348 {size=0);size=size; 1388
1349 int i,j,k;float s,t,u,max=0;float[] [] [] save2=new float 1389
1350 [105][105][105][3]; 1390
1351 for(s=0;s<1;s+=gap){for(t=0;t<1;t+=gap){if(s+t>1){continue;}u=1.0-s-t; 1391
1352 float sumx=0,sumy=0; 1392
1353 for(i=0;i<2;i++){for(j=0;j<2;i++)for(j=0;j<2;i++);sumx+=com2[i] 1393
1354 J[j][k]*power(s,i)*power(t,j)*power(u,k)+T[i][j][k][0]}]; 1394
1355 for(i=0;i<1;i++){for(j=0;j<2;i++)for(j=0;j<2;i++);sumy+=com2[i] 1395
1356 J[j][k]*power(s,i)*power(t,j)*power(u,k)+T[i][j][k][1]}]; 1396
1357 for(i=0;i<1;i++){for(j=0;j<2;i++)for(j=0;j<2;i++);sumz+=com2[i] 1397
1358 J[j][k]*power(s,i)*power(t,j)*power(u,k)+T[i][j][k][2]}]; 1398
1359 save2[int((s+1)*10]+int((t+1)*10)[int((u+1)*10)[0]]=sumx;save2[int( 1401
1360 (s+1)*10]+int((t+1)*10)[int((u+1)*10)[1]]=sumy;save2[int( 1402
1361 (s+1)*10]+int((t+1)*10)[int((u+1)*10)[2]]=sumz; 1403
1362 if(max*max*sumx*sumy*sumz*sumz>=max=sqrt(sum*sumx*sumy* 1404
1363 sumy*sumz*sumz)}]; 1405
1364 for(s=0;s<1;s+=gap){float savex=0,savey=0,savez=0;for(t=0;t<1;t+=gap){if(s 1406
1365 +t>1){continue;}u=1.0-s-t; 1407
1366 float k1,sumx=save2[int((s+1)*10]+int((t+1)*10)[int((u+1)*10) 1408
1367 *10][0],sumy=save2[int((s+1)*10]+int((t+1)*10)[int((u+1)*10) 1409
1368 *10][1],sumz=save2[int((s+1)*10]+int((t+1)*10)[int((u+1)*10) 1410
1369 *10][2]; 1411
1370 k1=sqrt(sumx*sumx+sumy*sumy+sumz*sumz)/max; point3D_RGB(sumx,sumy, 1412
1371 sumz,(int)(R*sqrt(sqrt(k1)*k1))+50,(int)(G*sqrt(sqrt(k1)*k1))+50,(int) 1413
1372 (B*sqrt(sqrt(k1)*k1))+50,5,siz); 1414
1373 if(s==0){savex=0,sumy=savey,sumz=0,sumz=0;} 1415
1374 else{line3D(savx,savx,savz,sumx,sumy,sumz,siz,(int)(R*sqrt(sqrt(k1)* 1416
1375 k1))+50,(int)(G*sqrt(sqrt(k1)*k1))+50,(int)(B*sqrt(sqrt(k1)*k1))+50, 1417
1376 +50,3);savx=savx,sumy=savy,sumz=savz,sumx=0,sumy=0,sumz=0;}} 1418
1377 for(t=0;t<1;t+=gap){float savex=0,savey=0,savez=0;for(u=0;u<1;u+=gap){if(u 1419
1378 +t>1){continue;}u=1.0-u-s; 1420
1379 float k1,sumx=save2[int((s+1)*10]+int((t+1)*10)[int((u+1)*10) 1421
1380 *10][0],sumy=save2[int((s+1)*10]+int((t+1)*10)[int((u+1)*10) 1422
1381 *10][1],sumz=save2[int((s+1)*10]+int((t+1)*10)[int((u+1)*10) 1423
1382 *10][2]; 1424
1383 k1=sqrt(sumx*sumx+sumy*sumy+sumz*sumz)/max; point3D_RGB(sumx,sumy, 1425
1384 sumz,(int)(R*sqrt(sqrt(k1)*k1))+50,(int)(G*sqrt(sqrt(k1)*k1))+50,(int) 1426
1385 (B*sqrt(sqrt(k1)*k1))+50,5,siz); 1427
1386 if(s==0){savex=0,sumy=savey,sumz=0,sumz=0;} 1428
1387 else{line3D(savx,savx,savz,sumx,sumy,sumz,siz,(int)(R*sqrt(sqrt(k1)* 1429
1388 k1))+50,(int)(G*sqrt(sqrt(k1)*k1))+50,(int)(B*sqrt(sqrt(k1)*k1))+50, 1430
1389 +50,3);savx=savx,sumy=savy,sumz=savz,sumx=0,sumy=0,sumz=0;}} 1431
1390 for(u=0;u<1;u+=gap){float savex=0,savey=0,savez=0;for(t=0;t<1;t+=gap){if(s 1432
1391 +t>1){continue;}u=1.0-u-s; 1433
1392 float k1,sumx=save2[int((s+1)*10]+int((t+1)*10)[int((u+1)*10) 1434
1393 *10][0],sumy=save2[int((s+1)*10]+int((t+1)*10)[int((u+1)*10) 1435
1394 *10][1],sumz=save2[int((s+1)*10]+int((t+1)*10)[int((u+1)*10) 1436
1395 *10][2]; 1437
1396 k1=sqrt(sumx*sumx+sumy*sumy+sumz*sumz)/max; point3D_RGB(sumx,sumy, 1438
1397 sumz,(int)(R*sqrt(sqrt(k1)*k1))+50,(int)(G*sqrt(sqrt(k1)*k1))+50,(int) 1439
1398 (B*sqrt(sqrt(k1)*k1))+50,5,siz); 1440
1399 if(t==0){savex=0,sumy=savey,sumz=0,sumz=0;} 1441
1400 else{line3D(savx,savx,savz,sumx,sumy,sumz,siz,(int)(R*sqrt(sqrt(k1)* 1442
1401 k1))+50,(int)(G*sqrt(sqrt(k1)*k1))+50,(int)(B*sqrt(sqrt(k1)*k1))+50, 1443
1402 +50,3);savx=savx,sumy=savy,sumz=savz,sumx=0,sumy=0,sumz=0;}} 1444
1403 for(s=0;s<1;s+=gap){float savex=0,savey=0,savez=0;for(u=0;u<1;u+=gap){if(s 1445
1404 +t>1){continue;}u=1.0-u-s; 1446
1405 float k1,sumx=save2[int((s+1)*10]+int((t+1)*10)[int((u+1)*10) 1447
1406 *10][0],sumy=save2[int((s+1)*10]+int((t+1)*10)[int((u+1)*10) 1448
1407 *10][1],sumz=save2[int((s+1)*10]+int((t+1)*10)[int((u+1)*10) 1449
1408 *10][2]; 1450
1409 k1=sqrt(sumx*sumx+sumy*sumy+sumz*sumz)/max; point3D_RGB(sumx,sumy, 1451
1410 sumz,(int)(R*sqrt(sqrt(k1)*k1))+50,(int)(G*sqrt(sqrt(k1)*k1))+50,(int) 1452
1411 (B*sqrt(sqrt(k1)*k1))+50,5,siz); 1453
1412 if(u==0){savex=0,sumy=savey,sumz=0,sumz=0;} 1454
1413 else{line3D(savx,savx,savz,sumx,sumy,sumz,siz,(int)(R*sqrt(sqrt(k1)* 1455
1414 k1))+50,(int)(G*sqrt(sqrt(k1)*k1))+50,(int)(B*sqrt(sqrt(k1)*k1))+50, 1456
1415 +50,3);savx=savx,sumy=savy,sumz=savz,sumx=0,sumy=0,sumz=0;}} 1457

```

```

1458 else if(itis==0){textSize(20); fill(255,0,0);text("Not Founded!",275,230)fill(0,0);}
1459 void istrue(){/\\{\nprintln(str);
1509
1460 if(str.equals("bezier3d"))==true){ans="Bezier3D(float x0,float y0,float z0
1510 ,float x1,float y1,float z1,float x2,float y2,float z2, float num,
1461 float siz);itis=1;
1511
1462 else if(str.equals("beziersurfacecontroliner44")=="true"){ans="
1511 Bezier_surface_control_line_rc44(float R,float G,float B,float siz)"
1512 ;itis=1;
1513
1463 else if(str.equals("beziersurfaceliner44")=="true"){ans="
1513 Bezier_surface_line_rc44(float R,float G,float B,float siz, float gap
1514 );itis=1;
1464 else if(str.equals("beziersurfacecontroliner33")=="true"){ans="
1514 Bezier_surface_control_line_rc33(float R,float G,float B,float siz)"
1515 ;itis=1;
1465 else if(str.equals("beziersurfaceliner33")=="true"){ans="
1515 Bezier_surface_line_rc33(float R,float G,float B,float siz, float gap
1516 );itis=1;
1466 else if(str.equals("beziertrianglecontrolline44")=="true"){ans="
1517 Bezier_triangle_control_line44(float R,float G,float B,float siz);
1518 itis=1;
1467 else if(str.equals("bezietrianglegeline44")=="true"){ans="
1519 Bezier_triangle_line44(float R,float G,float B,float siz, float gap)
1520 ;itis=1;
1468 else if(str.equals("bezietrianglecontrolline33")=="true"){ans="
1520 Bezier_triangle_control_line33(float R,float G,float B,float siz);
1521 itis=1;
1469 else if(str.equals("bezietrianglegeline33")=="true"){ans="
1522 Bezier_triangle_line33(float R,float G,float B,float siz, float gap)
1523 ;itis=1;
1470 else if(str.equals("checkrc44")=="true"){ans="checkrc44(int m,float x1,
1523 float y1,float z1,float x2,y2,float z2,float x3,y3,float z3
1471 else if(str.equals("checkrc33")=="true"){ans="checkrc33(int m,float x1,
1524 float y1,float z1,float x2,y2,float z2,float x3,y3,float z3
1472 else if(str.equals("check3d")=="true"){ans="check3D(float x,float y,float z
1525 ,float siz);itis=1;
1473 else if(str.equals("check3d3")=="true"){ans="check3D(float x,float y,float z
1526 ,float siz);itis=1;
1474 else if(str.equals("point3D")=="true"){ans="point3D(float x,float y,float z
1528 ,float siz);itis=1;
1475 else if(str.equals("point3drgb")=="true"){ans="point3D_RGB(float x,float y,
1529 float z,float R,float G,float B,float st,strokeWeight());itis=1;
1476 else if(str.equals("line3d")=="true"){ans="line3D(float x,float y,float z
1530 float w,float m,float n,float siz,float R,float G,float B,float
strokeWeight());itis=1;
1477 else if(str.equals("cuboid")=="true"){ans="Cuboid(float x0,float y0,float
1531 z0,float dr,int R,int G,int B,float siz,strokeWeight());itis=1;
1478 else if(str.equals("sphere")=="true"){ans="sphere3D(float x,float y,float z
1532 float R,float G,float B,float dr,float st,strokeSiz);itis=1;
1479 else{itis=0;}}
1533
1480 void showstring2(float x,float y){
1481 float z=y;if(XcodeLength==0){XcodeLength=1;}
1482 textSize(20);fill(0,0);text("lines",150,40);fill(0,0);text(XcodeLength
1535 ,210,40);text("NO SHIFT go back TAB",350,40);
1483 if(XcodeLength>20){XcodeLength=20;textSize(30);fill(255,0,0);text("1536
OVERFLOW",160,685);fill(0,0);textSize(20);}
1484 for(int i=1;i<21;i++){strokeWeight(0.1);
1485 if(i==XcodeLength){fill(255,0,0);stroke(255,0,0);fill(175,stroke(125);
rect(150,50+30*(i-1),1300,30);fill(255,0,0);stroke(255,0,0);}
1486 else{fill(0,0);stroke(0,fill(255);rect(150,50+30*(i-1)
1487 ,1300,30);fill(0,0);stroke(0);
strokeWeight(5);text(i,x-45,z-5);++30;}
1488 for(int i=1;i<21;i++){textSize(25);fill(0,0);stroke(0);text(win5[i],x,y);y
1489 ++30;}}
1540
1490 void console(){
1491 for(int i=0;i<5;i++){println(" ");}
1492 println("consolestart",consoleTemp);consoleTemp++;
1541 for(int i=1;i<10000;i++){
1493 if(search(i)==1){println("search",i,"checked!");}
1542 else{println("search",i,"failed!!");return;}println("consol end");
1543 consolCheck=1;return;}
1497
1498 int search(int k){
1499 String make="";
1500 if((win5[k].length()>8)&(win5[k].substring(0,8).equals("bezier3d")
1501 ==true){make="1541
print("Bezier3D(float x0,float y0,float z0,float x1,float y1,float
1542 z1,float x2,float y2,float z2, float num,float siz)";
1503 make=win5[k].substring(9,win5[k].length()-1);String[] makestr = split
1504 (make,' ');/\\{\nprintln(make);
1505 for(int i=0;i<1;i++){println(makestr[i]);}
1506 else if((win5[k].length()>28)&(win5[k].substring(0,28).equals("1549
beziersurfacecontroliner44")=="true"){print("1550
Bezier_surface_control_line_rc44(float R,float G,float B,float siz)
1551 ");
make=win5[k].substring(29,win5[k].length()-1);String[] makestr = 1551
split(make,' ');/\\{\nprintln(make);
1507 for(int i=0;i<4;i++){println(makestr[i]);}
1508 else if((win5[k].length()>21)&(win5[k].substring(0,21).equals("1552
beziersurfaceliner44")=="true"){print("1552
Bezier_surface_line_rc44(float R,float G,float B,float siz, float gap)
1553 ");
String[] makestr = split(make,' ');/\\{\nprintln(make);

```

```

1554     for(int i=0;i<9;i++){println(makestr[i]);}
1555     else if ((win5[k].length()>5)&&(win5[k].substring(0,5).equals("spere
1556     ")==true)){println("spere3D(float x,float y,float z,float R,float G,
1557     float B,float dr,float st,float siz");}
1558     make=win5[k].substring(6,win5[k].length()-1);
1559     String[] makestr = split(make,';');
1560     for(int i=0;i<9;i++){println(makestr[i]);}return 1;
1561   void consoledraw(){
1562     for(int k=1;k<10000;k++){
1563       String make="";float[] makefloat = new float[100];
1564       if((win5[k].length()>8)&&(win5[k].substring(0,8).equals("bezier3D")
1565       ==true){//"/"Bezier3D(float x0,float y0,float z0,float x1,float y1,
1566       float z1,float x2,float y2,float z2, float num, float siz)"}
1567       make=win5[k].substring(9,win5[k].length()-1);String[] makestr = split(
1568       make,';');
1569       for(int i=0;i<9;i+1)/{*println(makestr[i].substring(0,1));}*if(((
1570       makestr[i].substring(0,1).equals("-")))(makefloat[i]=-1)*floattry(
1571       makestr[i].substring(1,makestr[i].length()-1));
1572       else{makefloat[i]=floattry(makestr[i]);}for(int i=10;i<11;i+1){
1573       makefloat[i]=inttry(makestr[i]);
1574       Bezier3D(makefloat[0],makefloat[1],makefloat[2],makefloat[3],
1575       makefloat[4],makefloat[5],makefloat[6],makefloat[7],makefloat[8],
1576       makefloat[9],makefloat[10]);}
1577       else if ((win5[k].length()>28)&&(win5[k].substring(0,28).equals(""
1578       Bezier_surface_control_line_rc44"{
1579       make=win5[k].substring(29,win5[k].length()-1);String[] makestr =
1580       split(make,';');
1581       for(int i=0;i<4;i+1){/*println(makestr[i].substring(0,1));}*if(((
1582       makestr[i].substring(0,1).equals("-")))(makefloat[i]=-1)*floattry(
1583       makestr[i].substring(1,makestr[i].length()-1));
1584       else{makefloat[i]=floattry(makestr[i]);}Bezier_surface_control_line_rc44(
1585       makefloat[0],makefloat[1],makefloat[2],makefloat[3],
1586       makefloat[4],makefloat[5],makefloat[6],makefloat[7],makefloat[8],
1587       makefloat[9],makefloat[10]);
1588       else if ((win5[k].length()>21)&&(win5[k].substring(0,21).equals(""
1589       Bezier_surface_line_rc33"{
1590       make=win5[k].substring(22,win5[k].length()-1);String[] makestr =
1591       split(make,';');
1592       for(int i=0;i<4;i+1){/*println(makestr[i].substring(0,1));}*if(((
1593       makestr[i].substring(0,1).equals("-")))(makefloat[i]=-1)*floattry(
1594       makestr[i].substring(1,makestr[i].length()-1));
1595       else{makefloat[i]=floattry(makestr[i]);}Bezier_surface_line_rc33(
1596       makefloat[0],makefloat[1],makefloat[2],
1597       makefloat[3],makefloat[4]);
1598       else if ((win5[k].length()>27)&&(win5[k].substring(0,27).equals(""
1599       beziertriangle_control_line44"{
1600       make=win5[k].substring(28,win5[k].length()-1);String[] makestr =
1601       split(make,';');
1602       for(int i=0;i<4;i+1){/*println(makestr[i].substring(0,1));}*if(((

```



```

1399 else{makefloat[i]=floattry(makestr[i]);}
1400 point3Ddata(makefloat[0],makefloat[1],makefloat[2],makefloat[3]); 1797
1401 else if((win5[k].length_)>10)&&(win5[k].substring(0,10).equals(" 1798
1402 Point3Drgb"))==true{ 1799
1403 makewin5[k].substring(11,win5[k].length_));String[] makestr = 1800
1404 split(makestr,'.');// 1801
1405 for(int i=0;i<8;i++){if((makestr[i].substring(0,1).equals(".")){ 1801
1406 makefloat[i]=(i-1)*floattry(makestr[i].substring(1,makestr[i].length_); 1802
1407 (-1));} 1803
1408 else{makefloat[i]=floattry(makestr[i]);} 1804
1409 point3D_RGBdata(makefloat[0],makefloat[1],makefloat[2],makefloat[3], 1804
1410 makefloat[4],makefloat[5],makefloat[6],makefloat[7]);} 1805
1411 else if((win5[k].length_)>6)&&(win5[k].substring(0,6).equals(" 1805
1412 line3d"))==true{ 1806
1413 makewin5[k].substring(7,win5[k].length_)-1);String[] makestr = split 1806
1414 (makestr,'.');// 1807
1415 for(int i=0;i<11;i++){if((makestr[i].substring(0,1).equals(".")){ 1807
1416 makefloat[i]=(i-1)*floattry(makestr[i].substring(1,makestr[i].length_); 1808
1417 (-1));} 1808
1418 else{makefloat[i]=floattry(makestr[i]);} 1809
1419 line3Ddata(makefloat[0],makefloat[1],makefloat[2],makefloat[3], 1810
1420 makefloat[4],makefloat[5],makefloat[6],makefloat[7],makefloat[8], 1811
1421 makefloat[9],makefloat[10]);} 1811
1422 else if((win5[k].length_)>6)&&(win5[k].substring(0,6).equals(" 1812
1423 cuboid"))==true{ 1813
1424 makewin5[k].substring(7,win5[k].length_)-1);String[] makestr = split 1813
1425 (makestr,'.');// 1814
1426 for(int i=0;i<4;i++){if((makestr[i].substring(0,1).equals(".")){ 1815
1427 makefloat[i]=(i-1)*floattry(makestr[i].substring(1,makestr[i].length_); 1816
1428 (-1));} 1817
1429 else{makefloat[i]=floattry(makestr[i]);} 1820
1430 for(int i=4;i<7;i++){makefloat[i]=inttry(makestr[i]);} 1820
1431 for(int i=7;i<9;i++){if((makestr[i].substring(0,1).equals(".")){ 1819
1432 makefloat[i]=(i-1)*floattry(makestr[i].substring(1,makestr[i].length_); 1820
1433 (-1));} 1820
1434 else{makefloat[i]=floattry(makestr[i]);} 1821
1435 Cubodata(makefloat[0],makefloat[1],makefloat[2],makefloat[3],(int) 1821
1436 makefloat[4],(int)makefloat[5],(int)makefloat[6],makefloat[7], 1822
1437 makefloat[8]);} 1823
1438 else if((win5[k].length_)>5)&&(win5[k].substring(0,5).equals("spere 1823
1439 ")==true){ 1824
1440 makewin5[k].substring(6,win5[k].length_)-1);String[] makestr = split 1824
1441 (makestr,'.');// 1825
1442 for(int i=0;i<9;i++){if((makestr[i].substring(0,1).equals(".")){ 1825
1443 makefloat[i]=(i-1)*floattry(makestr[i].substring(1,makestr[i].length_); 1826
1444 (-1));} 1826
1445 else{makefloat[i]=floattry(makestr[i]);} 1827
1446 for(int i=9;i<17;i++){spere3Ddata(makefloat[0], 1827
1447 makefloat[1],makefloat[2],makefloat[3],makefloat[4],makefloat[5], 1827
1448 makefloat[6],makefloat[7],makefloat[8]);} 1828
1449 else if((win5[k].length_)>5)&&(win5[k].substring(0,17).equals(" 1828
1450 examplebezierrect"))==true){example_Bezier_rectdata();} 1828
1451 else if((win5[k].length_)>5)&&(win5[k].substring(0,16).equals(" 1829
1452 examplebeziertri"))==true){example_Bezier_tridata();} 1829
1453 else if((win5[k].length_)>5)&&(win5[k].substring(0,17).equals(" 1830
1454 examplebezierline"))==true){example_Bezier_linedata();} 1830
1455 data[count]="#length": "+Integer.toString(count);count++; 1831
1456 for(int i=count;i<10000000;i++){data[i]="#";} 1832
1457 saveString("data\\bvffile.txt",data); 1833
1458 data2[count2]="#length": "+Integer.toString(count2);count2++; 1834
1459 for(int i=count2;i<1000000;i++){data2[i]="#";} 1835
1460 saveString("data\\vertex.txt",data2); 1836
1461
1462 void showstring3(){ 1837
1463 stroke(0);fill(0); 1837
1464 textSize(20);text("progress",150,125); 1838
1465 for(int i=0;i<win4length;i++){ 1838
1466 textSize(20); 1839
1467 text(win4[i],175,180+i*30);} 1839
1468
1469 void win4clear(){ 1840
1470 for(int i=0;i<win4length;i++){win4[i]="#";} 1841
1471 win4length=0;} 1841
1472
1473 void win4push(String t){ 1842
1474 win4[win4length]="#"; 1842
1475 win4length++;} 1842
1476
1477 void parsefile0_ {BufferedReader reader = createReader("data\\textfile\\sphere.obj");String line = null;String a=null; 1843
1478 try {while ((line = reader.readLine()) != null) {String[] pieces= 1843
1479 split(line,'.');// 1844
1480 if(pieces[0].equals("v#"))==true{(obj[row][0]=Size*float(pieces[2]);obj[ 1844
1481 row][1]=Size*float(pieces[4]);obj[row][2]=Size*float(pieces[3]; // 1844
1482 sphere obj[row][1],obj[row][2] change 1845
1483 if(obj[row][2]-az>zmx){zmx= 1845
1484 -obj[row][2]-az;zmw=obj[row][2]-az;}if(obj[row][2]-az>zmx){zmx= 1845
1485 -obj[row][2]-az;row++;} 1845
1486 if(pieces[0].equals("vn#"))==true{vervec[row][0]=10*float(pieces[1]); 1846
1487 vervec[row][1]=10*float(pieces[2]);vervec[row][2]=10*float(pieces 1846
1488 [3];row++;} 1847
1489 if(pieces[0].equals("f#"))==true{for(int i=1;i<5;i++)a={pieces[i]; 1847
1490 String[] pieces2=a.split('/');//v vt vt 1848
1491 index[row][i-1]=int(pieces[0]);index[row][i-1][1]=int( 1848
1492 pieces2[1]);index[row][i-1][2]=int(pieces2[2]); 1849
1493 row++;}reader.close();} catch (IOException e) {e.printStackTrace() 1849
1494 ;}} 1850
1495
1496 void parsefile2_ {BufferedReader reader = createReader("data\\textfile\\ 1850

```

```

1851 /*float lumminosity1=1/(1+distance1*distance1),lumminosity2=1/(1+
1852 distance2*distance2),lumminosity3=1/(1+distance3*distance3),
1853 lumminosity4=1/(1+distance4*distance4)*/ 1895
1854 float lumminosity1=1/(1+length1*length1),lumminosity2=1/(1+length2*
1855 length2),lumminosity3=1/(1+length3*length3),lumminosity4=1/(1+
length4*length4); 1898
1856 /float finalcolorR=255,finalcolorG=0,finalcolorB=0,finalcolor2R=
1857 255,finalcolor2G=0,finalcolor2B=0,finalcolor3R=255,finalcolor3G=0,
1858 finalcolor3B=0,finalcolor4R=255,finalcolor4G=0,finalcolor4B=0;*/ 1900
1859 float finalcolorR=RMaterial[Ka][0],finalcolorG=GMaterial[Ka][1],
1860 finalcolorB=BMaterial[Ka][2]; 1902
1861 float finalcolor1R=RMaterial[Ka][0]+lightpower*lightcolorR* 1904
1862 lambertfactor1*lumminosity1),finalcolor1G=GMaterial[Ka][1]+ 1905
1863 lightpower*lightcolorG*lambertfactor1*lumminosity1),finalcolor1B= 1906
1864 BMaterial[Ka][2]+lightpower*lightcolorB*lambertfactor1*lumminosity1 1907
1865 ; 1908
1866 float finalcolor2R=RMaterial*[Ka][0]+lightpower*lightcolorR* 1909
1867 lambertfactor2*lumminosity2),finalcolor2G=GMaterial*[Ka][1]+ 1910
1868 lightpower*lightcolorG*lambertfactor2*lumminosity2),finalcolor2B= 1911
1869 BMaterial*[Ka][2]+lightpower*lightcolorB*lambertfactor2*lumminosity2 1912
1870 ; 1913
1871 float finalcolor3R=RMaterial*[Ka][0]+lightpower*lightcolorR* 1914
1872 lambertfactor3*lumminosity3),finalcolor3G=GMaterial*[Ka][1]+ 1915
1873 lightpower*lightcolorG*lambertfactor3*lumminosity3),finalcolor3B= 1916
1874 BMaterial*[Ka][2]+lightpower*lightcolorB*lambertfactor3*lumminosity3 1917
1875 ; 1918
1876 float finalcolor4R=RMaterial*[Ka][0]+lightpower*lightcolorR* 1919
1877 lambertfactor4*lumminosity4),finalcolor4G=GMaterial*[Ka][1]+ 1920
1878 lightpower*lightcolorG*lambertfactor4*lumminosity4),finalcolor4B= 1921
1879 BMaterial*[Ka][2]+lightpower*lightcolorB*lambertfactor4*lumminosity4 1922
1880 ; 1923
1881 rect3D_RGB(x1,y1,z1,finalcolor1R,finalcolor1G,finalcolor1B,x2,y2,z2,
1882 finalcolor2R,finalcolor2G,finalcolor2B,x3,y3,z3,finalcolor3R,
1883 finalcolor3G,finalcolor3B,x4,y4,z4,finalcolor4R,finalcolor4G,
1884 finalcolor4B,finalcolorR,finalcolorG,finalcolorB,2,siz); 1924
1885 fill(0);stroke(0); 1925
1886 void rect3D_RGB(float x1,float y1,float z1,float R1,float G1,float B1,float
1887 x2,float y2,float z2,float R2,float G2,float B2,float x3,float y3,
1888 R3,float R3,float G3,float B3,float x4,float y4,float z4,float
1889 R4,float G4,float B4,float R,float G,float B,float st,float siz){ 1929
1890 if(openLoop==0){ 1929
1891 {ffend}[0]=x-ax;{ffend}[1]=y1-ay;{ffend}[2]=z1-az;{ffend}[3]=x2-ax;if({ffend}
1892 [4]==y2-ay;{ffend}[5]==z2-az;{ffend}[6]==x3-ax;{ffend}[7]==y3-ay;{ffend}
1893 [8]==z3-az;ffend++; 1928
1894 {ffend}[0]==x-ax;{ffend}[1]==y2-ay;{ffend}[2]==z2-az;{ffend}[3]==x3-ax;if({ffend}
1895 [4]==y3-ay;{ffend}[5]==z3-az;{ffend}[6]==x4-ax;{ffend}[7]==y4-ay;{ffend}
1896 [8]==z4-az;ffend++); 1931
1897 if(siz!=size){float tri=new float [10]; 1932
1898 check3D(x1,y1,z1,siz);tri[0]=950+0[A];tri[1]=400+A[1];check3D(x2,y2,z2,
1899 siz);tri[2]=950+0[A];tri[3]=400+A[1];check3D(x3,y3,z3,siz);tri
1900 [4]=950+0[A];tri[5]=400+A[1];check3D(x4,y4,z4,siz);tri[6]=950+[A] 1935
1901 tri[7]=400+A[1]; 1936
1902 point3D_RGB(x1-ax,y1-ay,z1-az,R1,G1,B1,st,siz);point3D_RGB(x2-ax,y2-ay,
1903 z2-az,R2,G2,B2,st,siz);point3D_RGB(x3-ay,y3-az,z3-az,R3,G3,B3,st,siz);
1904 point3D_RGB(x4-ay,y4-az,z4-az,R4,G4,B4,st,siz); 1938
1905 Bpoint3D(x1-ax,y1-ay,z1-az);Bpoint3D(x2-ax,y2-ay,z2-az);Bpoint3D(x3-ax-
1906 y3-ay,z3-az);Bpoint3D(x4-ay,y4-ay,z4-az); 1940
1907 fill(0);strokeWeight(5); 1941
1908 void exisubfunction(){ 1942
1909 for(int i=0;i<10000;i++){p[i]=new vector();} 1943
1910 for(int i=0; i<10; i++) for(int j=0; j<10; j++) {p[10*i+j]=new vector( 1944
1911 ((2*i+1),(2*j-9),sqrt((200-(2*i+9)*(2*i-9)-(2*j-9)*(2*j-9)));u[10*i+j]
1912 =0.5; v[10*i+j]=0.5; 1945
1913 b[0][0]=new vector(-150, -150, 0); b[2][0]=new vector(150, -150, 0);b[4]
1914 [0][2]=new vector(-150, 150, 0); b[2][2]=new vector(150, 150, 0);b[4]
1915 [0][1]=new vector(0, -150, 300); b[1][0]=new vector(0, -150, 300);b[4]
1916 [0][1]=new vector(150, 0, 300); b[1][2]=new vector(0, 150, 300);b[5]
1917 [1][1]=new vector(); 1952
1918 for(int i=0; i<20; i++){Find_b1(); Rearrange();} 1953
1919 return;} 1954
1920 void showexample(){ 1955
1921 check3r33(0,b[0][0].vectori(0),b[0][0].vectori(1),b[0][0].vectori(2) 1957
1922 b[0][1].vectori(0),b[0][1].vectori(1),b[0][1].vectori(2),b[0][2] 1958
1923 vectori(0),b[0][2].vectori(1),b[0][2].vectori(2)); 1959
1924 check3r33(1,b[1][0].vectori(0),b[1][0].vectori(1),b[1][0].vectori(2) 1960
1925 b[1][1].vectori(0),b[1][1].vectori(1),b[1][1].vectori(2),b[1][2] 1961
1926 vectori(0),b[1][2].vectori(1),b[1][2].vectori(2)); 1962
1927 check3r33(2,b[2][0].vectori(0),b[2][0].vectori(1),b[2][0].vectori(2) 1963
1928 b[2][1].vectori(0),b[2][1].vectori(1),b[2][1].vectori(2),b[2][2] 1964
1929 vectori(0),b[2][2].vectori(1),b[2][2].vectori(2)); 1965
1930 Bezier_surface_line_rc33(255, 0, 255, size, 0.05); 1966
1931 Bezier_surface_control_line_rc33(255, 246, 142, size);} 1967
1932 void show(){ 1968
1933 for(int i=0;i<showend;i++){ 1969
1934 //if(sh[i]==0)point3D_RGB(show[i][0],show[i][1],show[i][ 1970
1935 [1][2],255,0,0.3,size); 1971
1936 if(sh[i]==1)point3D_RGB(show[i][0],show[i][1],show[i][ 1972
1937 [1][2],255,0.25,0.3,size); 1973
1938 if(sh[i]==2)point3D_RGB(show[i][0],show[i][1],show[i][ 1974
1939 [1][2],255,150,0.3,size); 1975
1940 if(sh[i]==3)point3D_RGB(show[i][0],show[i][1],show[i][ 1976
1941 [1][2],150,0,155,0.3,size); 1977
1942 if(sh[i]==4)point3D_RGB(show[i][0],show[i][1],show[i][ 1978
1943 [1][2],0,255,3,size); 1979

```

<pre> 1980 checkt(1, 0, 1, 75, -75, 50); 1981 checkt(0, 2, 0, 104, 115, 92); 1982 checkt(0, 1, 1, 100, -104, 93); 1983 checkt(0, 0, 2, -231, -131, 31);} 1984 1985 void checktr33_sample(){ 1986 checkt(2, 0, 0, -200, -200, 200); </pre>	<pre> 1987 checkt(1, 1, 0, 200, 0, 200); 1988 checkt(1, 0, 1, 0, 200, 200); 1989 checkt(0, 2, 0, 200, -200, -200); 1990 checkt(0, 1, 1, 200, 200, 0); 1991 checkt(0, 0, 2, -200, 200, -200);} </pre>
---	--

IV. Conclusion

이 연구에서 임의의 3차원 곡선을 2차 베지어 곡선을 통해 근사할 때에는 3차원 곡선을 조각별로 나누고, 각각의 조각에서 적절한 조절점을 정의하는 방법을 이용함으로써 어떤 곡선이든 충분히 근사 가능함을 알아내었다. 그리고 두 곡선, 즉 처음 곡선과 끝 곡선이 주어졌을 때, 연속적인 변화를 통해 곡선의 움직임을 줄 수 있는 방법을 제시하였다.

각각의 베지어 곡선을 구성하는데 있어서 세 개의 점의 좌표만이 요구되기 때문에 곡선을 저장하는데 있어서 필요한 용량이 효율적으로 감소된다. 또한 본 연구의 방법을 통해서 3차원 곡선, 곡면을 자유자재로 다룰 수 있기 때문에 VR, 휴로그램 등 미래에 상용화될 것이라고 기대되는 기술들을 구현함에 있어 효과적인 역할을 할수 있을 것이라고 예상된다.

3D 근사의 경우에는 obj파일로 정점들의 집합과 곡면을 이루는 다각형 면들이 주어졌을 때 베지어 곡면을 이용해 원 곡면을 근사한다. 이를 위해 정점들을 원통형으로 분할하며, 분할된 각 영역에 대해 베지어 곡면의 성질과 Möller-Trumbore intersection algorithm, 최소 제곱법, 뉴턴-랩슨 방법을 이용해 베지어 곡면을 구성하는 조절점을 얻는다. 더욱이 제시된 방법에 따라 연구 대상으로 하는 곡면들이 모두 충분히 근사될 수 있음을 보였다.

실제로 반지름의 길이가 64인 구면에 연구에서 제시한 방법을 적용했을 때, 실행 시간 1분 이내에 오차함수 $\mu < 10$ 을 얻었고 곡면을 저장하는 메모리는 200배 이상 감소 되었다.

베지어 곡면은 삼각형 분할에 비해 더 효율적이며, 적은 메모리를 요구한다. 그렇기에 본 연구는 VR, AR 등 곡면에 대한 그래픽 기술을 다루는 분야에서의 활용이 기대되며 특히 CAGD 분야에 있어 의미있는 결과이다. 또한 기존에 3D 모델을 저장하기 위해 자주 사용되는 obj파일을 변환시키는 연구이기에 3D 모델링이 필요한 보다 다양한 분야와 접목시킬 수

있다.

그리고 추가적인 후속연구로 2D,3D 근사 기법을 통해 어떤 색깔 있는 2D,3D 곡면을 더 부드럽게 표현할 수 있었고 모델만 저장하는 형식보다는 아니지만 기존의 RGB 저장 형식을 개선할 수 있었다. 모델 근사 뿐만 아니라 이에 대한 색깔까지 저장할 수 있으므로 더 많은 쪽에서 활용을 기대할 수 있다.

References

- [1] Ji-wung Choi, Renwick Curry, Babriel Elkaim (2008) “Path Planning Based on Bezier Curve for Autonomous Ground Vehicles”, WCECS 2008.
- [2] Hazewinkel, Michiel (1997) “Encyclopaedia of Mathematics: Supplement. 1”, Springer Science & Business Media, pp. 119.
- [3] Seon-Hong Kim, Young Joon Ahn (2007) “Approximation of circular arcs by quartic Bezier curves”, Computer-Aided Design 39, pp. 490-493.
- [4] A. R. Forrest (1972) “Interactive interpolation and approximation by Bezier polynomials”, The Computer Journal, Vol 15, Issue 1, pp. 71-79.
- [5] 문성룡, 문홍진, 송의남, 김종교 (1985) “Quadratic Bezier Polynomial 과 B-spline Function을 이용한 Curve Fitting 및 Computer Graphic Animation 에 관한 기초연구”, 대학 전자공학회 학술대회, pp. 321-327.
- [6] Munkres, James (1999) Topology(2nd ed) Prentice Hall, pp.280-281.
- [7] Farin, G. E., & Farin, G. (2002). *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann.
- [8] Möller, T., & Trumbore, B. (1997). Fast, minimum storage ray-triangle intersection. *Journal of graphics tools*, 2(1), 21-28.
- [9] Munkres, J. (2014). *Topology*. Pearson Education.
- [10] Lifton, J. J., Liu, T., & McBride, J. (2021). Non-linear least squares fitting of Bézier surfaces to unstructured point clouds. *AIMS Mathematics*, 6(4), 3142-3159.

감사의 글

먼저 지난 3년 동안 저에게 헤아릴 수 없는 많은 가르침을 주신 경기과학고등학교 선생님들께 깊은 감사의 말씀을 올립니다. 항상 부족한 제자였던 저를 넓은 아량으로 감싸주시고 이끌어 주셨기에 제가 이 만큼 성장할 수 있었습니다. 선생님들의 훌륭한 가르침에 마음을 가득 담아 존경을 표하며 감사드립니다. 선생님들의 가르침을 마음 깊이 새기고 앞으로 나아감으로써 은혜에 보답할 수 있도록 하겠습니다.

기초 R&E와 심화 R&E를 지도해주신 김소연 선생님께도 깊은 감사의 말씀을 올립니다. 2년 동안 연구를 진행하면서 힘들고 어려운 일들이 있을 때마다 많은 조언과 가르침을 주셨기에 연구를 무사히 마무리할 수 있었습니다. 늘 바쁘신 와중에도 이 논문이 처음부터 완성에 이르기까지 물심양면 도와주시고 항상 응원해 주셔서 감사드립니다. 그리고 바쁘신 와중에도 이 논문에 관심을 가져주시고 논문의 심사를 맡아주신 김소연 선생님과 박병철, 서은아 심사위원님께 감사의 말씀을 올립니다.

3년이라는 긴 시간 동안 동고동락한 경기과학고등학교의 선배, 후배, 학우들에게도 감사의 말씀을 드립니다. 선배, 후배, 학우들이 있었기에 힘들 때마다 함께 의지하고 헤쳐나가며 학교생활을 즐겁고 행복하게 할 수 있었습니다. 경기과학고등학교에서 우리가 함께 보낸 소중하고 뜻깊은 고등학교 생활이 앞으로 다가올 제 삶에 훌륭한 밑거름이 될 것이라고 믿습니다. 모두 행복하고 즐거운 삶을 살아가길 진심으로 바라며, 앞으로 하는 모든 일들이 축복과 함께하길 기원합니다.

그리고 사랑하는 가족에게도 감사의 말씀을 올립니다. 가장 힘든 순간마다 언제나 저를 지지해 주시고 격려해주신 부모님께 감사드립니다. 항상 저에 대해 함께 고민해 주시고 지

혜 담긴 말씀들로 저의 갈 길을 열어주신 부모님과 동생인 저를 아끼고 지지해 준 누나에게 진심어린 사랑을 전합니다.

지금까지 너무나 많은 분들이 저에게 큰 도움을 주셨기에 이 짧은 감사의 글로 제 마음을 다 표현할 수는 없지만, 지금까지 함께 해주신 모든 분들께 진심으로 감사의 말씀을 올리며, 저에게 보내주신 응원과 사랑을 마음 깊이 새기고 앞으로도 최선을 다하는 삶을 살아가겠습니다. 감사합니다.

연 구 활 동

- 2020 국제수학교육학술대회 R&E 포스터 논문 발표 대회 장려상 수상
- 2021 심화 R&E 우수상(수학 부문)
- 2021 영재학교 우수 R&E 공동발표회 우수상
- 2021 삼성휴먼테크논문대상 (수학/전산부문) 동상