***기초 컴퓨터 그래픽스 – 프로그래밍 숙제 1 (Ver. 1.0)***

***OpenGL API 함수를 사용한 2차원 기하 변환 연습***

20111633 신광수

과제 명: 신시티(Shin City)

과제 소개: 심시티라는 게임은 많이 아실 듯 하다. 이는 심시티의 아류작, 신시티이다. 게임 신시티는 주어진 부지 안에 집을 세우고, 그 집에서 거주하는 주민들의 차량의 움직임을 통해 도시의 디자인을 진행하는 게임이다. 신시티에서는 총 7개의 물체가 있다. 7개의 물체에 대한 소개 및 활용도는 다음의 표와 같다.



|  |  |  |
| --- | --- | --- |
| 물체 | 소개 | 사용법 |
|  | 신시티 상공에서 타원을 그리며 회전하며 치안을 담당하는 **비행기**이다. | 없음 |
|  | 신시티의 괴짜 과학자 신모씨가 만든 **우주선**이다. 괴짜가 만든 만큼, 모든 움직임과 크기는 그의 맘대로, 조정할 수 없다. 다만 불법으로 만든 탓에 비행기를 만나면 도망친다. | ‘R’키: 우주선이 다른 위치로 워프하여 새로운 위치에서 나타난다. |
|  | 신시티의 주민들이 거주하는 **집**. 집은 신시티 내에 최대 10개만 존재한다. 새로운 집이 등장하면 기존에 가장 오래된 집은 부셔진다. | ‘왼쪽’클릭: 지도 위에 새로운 집을 만들어준다. |
|  | 신시티의 주민들이 이동을 위해 사용하는 **자동차**. 이 차는 항상 도시의 외곽을 향해 영점에서 집의 각도로 나아간다. 집이 외곽으로 모두 나가면 다시 집에서 차량 한 대가 나간다. | 없음 |
|  | 신시티의 언론인의 **차량**. 비행기와 우주선에 대해 취재를 한다. 공평한 보도를 위해 비행기와 우주선의 중간에서 달린다. | 없음 |
|  | 신시티의 유일한 오락거리 **주점**이다. 주점은 집이 두 채 이상 있을 때부터 영업을 시작하며, 그 장소는 집들의 정 가운데에 위치한다. | 없음 |
|  | 신시티의 언론사가 주최하는 페스티벌 기념 **티셔츠**이다. 이 기념 티셔츠는 언론인의 차량에서 배부된다. | ‘F’키: 차량에서 360도 방향으로 축제 기념 옷을 뿌린다. |

기타 키 및 물체에 대한 설명은 다음과 같다.

|  |  |  |
| --- | --- | --- |
| 종류 | 소개 | 내용 / 사용법 |
| **[물체]** | **축 (Axis)** | 신시티의 원점을 알려주고, x, y 방면을 알려주는 이정표 |
| **[물체]** | **빨간 선** | 언론인의 차량과 주점을 연결하는 통신선 |
| **[입력]** | **‘오른쪽’ 클릭** | 괴짜 과학자 신모씨의 장난으로, 위의 빨간 선을 기준으로 모든 집이 대칭이동되며, 자동차 또한 새로 추가된 집 위치에 맞게 추가되어 나온다. |
| **[입력]** | **‘+’키** | 신시티의 전체적인 속도를 빠르게 한다. |
| **[입력]** | **‘-’키** | 신시티의 전체적인 속도를 느리게 한다. |
| **[입력]** | **‘Space’키** | 신시티을 일시정지/재생 한다. |
| **[입력]** | **‘C’키** | 신시티의 Crazy 모드를 활성화/비활성화한다. Crazy 모드란, 이름 그대로 미친 모드이다. 언론인의 차량에서는 끊임없이 티셔츠가 뿌려지고, 각 집에서 차량들이 무한대로 나온다. |
| **[입력]** | **‘ESC’키** | 신시티를 종료한다. |

위에서 기술한 움직임들은 창의 크기에 상관 없이 작동하도록 설정하였다.

이 아래는 기술적인 내용을 위한 문서이다.

주어진 6개의 물체 이외에 만든 ‘우주선’은 다음과 같이 모델링 되었다.

GLfloat spaceship\_body[4][2] = { { 10.0, 17.0 }, { -10.0, 17.0 }, { -10.0, -17.0 }, { 10.0, -17.0 } };

GLfloat spaceship\_window[8][2] = { { 5.0, 0 }, { five\_div\_root2, five\_div\_root2 }, { 0.0, 5.0 }, { -five\_div\_root2, five\_div\_root2 }, { -5.0, 0.0 }, { -five\_div\_root2, -five\_div\_root2 }, { 0.0, -5.0 }, { five\_div\_root2, -five\_div\_root2 } };

GLfloat spaceship\_head[6][2] = { { 10.0, 17.0 }, { 10.0, 18.0 }, { 2.0, 23.0 }, { -2.0, 23.0 }, { -10.0, 18.0 }, { -10.0, 17.0 } };

GLfloat spaceship\_fire\_center[3][2] = { { 3.0, -17.0 }, { 0, -25.0 }, { -3.0, -17.0 } };

GLfloat spaceship\_fire\_left[3][2] = { { -3.0, -17.0 }, { -6.5, -23.0 }, { -10.0, -17.0 } };

GLfloat spaceship\_fire\_right[3][2] = { { 3.0, -17.0 }, { 6.5, -23.0 }, { 10.0, -17.0 } };

GLfloat spaceship\_wing\_left[3][2] = { { -10.0, 0 }, { -10.0, -10.0 }, {-20.0, -10.0} };

GLfloat spaceship\_wing\_right[3][2] = { { 10.0, 0 }, { 10.0, -10.0 }, {20.0, -10.0} };

GLfloat spaceship\_color[8][3] = {

{ 255 / 255.0f, 255 / 255.0f, 255 / 255.0f },

{ 0 / 255.0f, 102 / 255.0f, 204 / 255.0f },

{ 32 / 255.0f, 32 / 255.0f, 32 / 255.0f },

{ 205 / 255.0f, 0 / 255.0f, 0 / 255.0f },

{ 204 / 255.0f, 0 / 255.0f, 0 / 255.0f },

{ 204 / 255.0f, 0 / 255.0f, 0 / 255.0f },

{ 32 / 255.0f, 32 / 255.0f, 32 / 255.0f },

{ 32 / 255.0f, 32 / 255.0f, 32 / 255.0f }

};

우주선의 랜덤한 움직임을 위해 랜덤 함수를 직접 만들어 구현하였다. XOR 연산을 통한 랜덤 수 생성 함수는 다음과 같다.

uint64\_t randXORseed[2] = { 0, (uint64\_t)time(NULL) };

uint64\_t randXOR(void) {

uint64\_t x = randXORseed[0];

uint64\_t const y = randXORseed[1];

randXORseed[0] = y;

x ^= x << 23; // a

randXORseed[1] = x ^ y ^ (x >> 17) ^ (y >> 26); // b, c

return randXORseed[1] + y;

}

차량과 옷. 우주선이 경계선을 나갔는지에 대한 확인을 하기 위한 함수는 다음과 같다.

bool window\_border\_check(glm::vec2 a) {

if (abs(a.x) >= win\_width / 2) return true;

if (abs(a.y) >= win\_height / 2) return true;

return false;

}

bool window\_border\_check(glm::vec2 a, int \*direction) {

if (abs(a.x) >= win\_width / 2) {

if (a.x < 0) \*direction = 'W';

else \*direction = 'E';

return true;

}

if (abs(a.y) >= win\_height / 2) {

if (a.y < 0) \*direction = 'S';

else \*direction = 'N';

return true;

}

\*direction = ' ';

return false;

}

또한 비행기와 우주선의 충돌을 감지하는 함수 또한 필요하여 제작하였다.

#define COLLISION\_THRESHOLD 20

bool collision\_check(glm::vec2 a, glm::vec2 b) {

if (abs(a.x - b.x) >= COLLISION\_THRESHOLD) return false;

if (abs(a.y - b.y) >= COLLISION\_THRESHOLD) return false;

return true;

}

bool collision\_check(glm::vec2 a, float size\_a, glm::vec2 b, float size\_b) {

if (abs(a.x - b.x) >= COLLISION\_THRESHOLD + (size\_a + size\_b) \* 1.5) return false;

if (abs(a.y - b.y) >= COLLISION\_THRESHOLD + (size\_a + size\_b) \* 1.5) return false;

return true;

}

이런 다양한 함수들을 기반으로 하여, 키 입력과 클릭에 따른 다양한 변수의 변화를 통해 위와 같은 프로그램을 만들 수 있었다.

본 신시티의 개발로 다음과 같은 내용을 습득할 수 있었다.

1. 2차원 아핀 변환에 대한 이해도 증가
2. 물체 모델링에 대한 기본적 지식 습득
3. 축과 물체의 위치에 대한 내용 습득
4. OpenGL의 마우스 및 키보드 입력에 대한 처리 과정 습득
5. OpenGL의 timer의 구조에 대한 기초적 내용 이해
6. (코드에는 구현하지 못했지만) 효과적인 물체 이동 방법에 대한 고찰

사용법에 대한 내용은 초기에 올라오는 콘솔 창에도 동시에 기재되어 있습니다. 참고 바랍니다.

