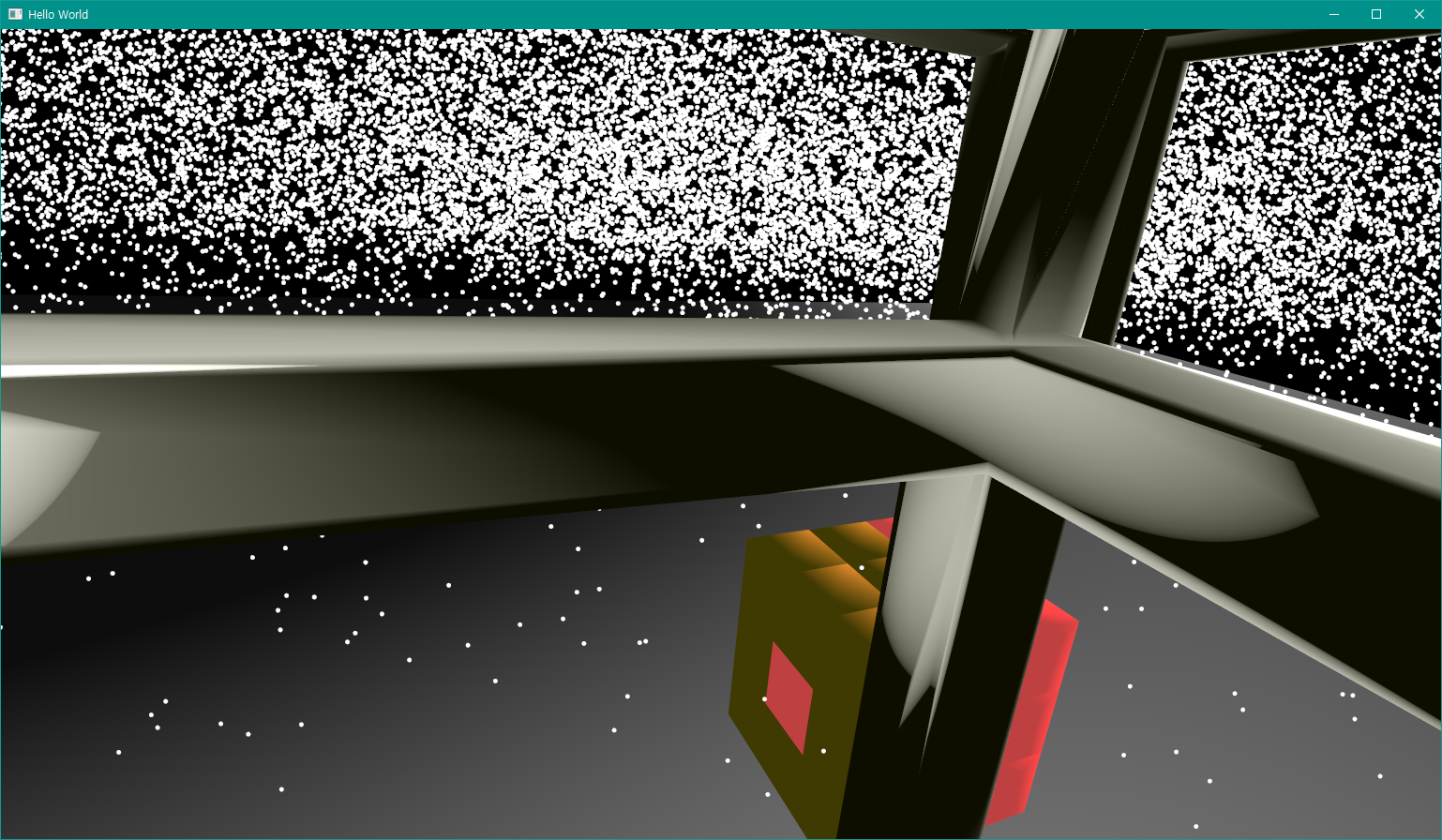
Game Development Report

# 게임 배경



눈내리는 어느 공간에 퍼즐만 덩그러니 놓여있습니다. 플레이어는 철장에 갖혀 있는데 퍼즐을 풀기 전까지는 일정 범위내에서 벗어날 수가 없습니다. 퍼즐을 풀면 철장에서 벗어나 자유롭게 움직일수 있습니다.

# 게임 키

게임을 시작할 때는 마우스 오른쪽 버튼을 누르면 움직일수 있습니다. 기본 키는 W A S D 로 앞뒤 양옆으로 움직일 수 있고 SPACE 를 누르면 일정기간 위로 오를 수 있지만 가만히 있으면 자동으로 낙하 됩니다.

큐브를 돌리는 키로는 R, F, T, G, Y, H, U, J, I, K, O, L 키로 순서대로 위쪽 회전 위반대방향 회전, 아래 회전, 아래 반대방향회전 식으로 위 아래 왼쪽 오른쪽 앞 뒤 실질적으로 키보드에서 순서대로 R T Y U I O 만으로 게임을 풀 수있고 아래의 키들은 반대 방향으로 회전시켜주는 용도 입니다.

# 주요 기능 코드

|  |
| --- |
| 큐브 돌리기 |
| void updatePhongSurface(const StaticTriangularSurface& surface, int num)  {    if (num==0) //위 회전  {  glm::vec3 position = glm::vec3(-2, 4.5, -4);  for (int i = 0; i < surface.vertex\_positions\_.num\_elements\_; i++) {  surface.vertex\_positions\_.values\_[i] -= position;  float radius = 90.0f;  float tsin = sin(glm::radians(radius));  float tcos = cos(glm::radians(radius));  float temp = surface.vertex\_positions\_.values\_[i].x\_;  surface.vertex\_positions\_.values\_[i].x\_ = surface.vertex\_positions\_.values\_[i].x\_\*tcos - surface.vertex\_positions\_.values\_[i].z\_\*tsin;  surface.vertex\_positions\_.values\_[i].z\_ = temp\*tsin + surface.vertex\_positions\_.values\_[i].z\_\*tcos;  surface.vertex\_positions\_.values\_[i] += position;  }  }  …  위 함수에서 어떤 회전을 해야할지 num변수로 받아와 회전시켜줍니다. |
| 눈 내리기 |
| 기존 파티클 시스템을 넓게 이용하였습니다. |
| 움직이기 |
| if (start\_move) {//마우스 우클릭  //카메라 따라다니기  glm::vec3 temp = cameraPos;  process\_key(cameraPos, v);  glm::vec3 dif = cameraPos - temp;  for (int i = 0; i < surface.vertex\_positions\_.num\_elements\_; i++) {  surface.vertex\_positions\_[i] = glm::mat4(1)  \*glm::translate(glm::vec3(dif))  \*glm::vec4(surface.vertex\_positions\_[i].x\_,  surface.vertex\_positions\_[i].y\_,  surface.vertex\_positions\_[i].z\_,  1.0f);  }  //카메라 따라 회전  for (int i = 0; i < surface.vertex\_positions\_.num\_elements\_; i++) {  surface.vertex\_positions\_[i] = glm::mat4(1)  \*glm::translate(cameraPos)  \*glm::mat4(glm::vec4(cos(glm::radians(dif\_yaw)),0,sin(glm::radians(dif\_yaw)),0), glm::vec4(0, 1, 0, 0),  glm::vec4(-sin(glm::radians(dif\_yaw)), 0, cos(glm::radians(dif\_yaw)), 0),  glm::vec4(0, 0, 0, 1) )  \*glm::translate(-cameraPos) \*glm::vec4(surface.vertex\_positions\_[i].x\_,  surface.vertex\_positions\_[i].y\_, surface.vertex\_positions\_[i].z\_,  1.0f);  }  dif\_yaw = 0.0f;  }  회전은 마우스 함수에서 차이만큼 받아와 옮겨 주었고 철장이 카메라 따라다니기는 카메라 위치로 철장을 움직여주어 구현하였습니다. |

# 향후 계획

현재는 텍스쳐를 제대로 다루지 못하여 완전한 루빅스 큐브를 만들지 못하였지만 이후 완벽한 큐브를 만들 것 입니다.