게임 프로그래밍

HW4

06/12/19

화18:00

컴퓨터공학부 201202914 김건우

목차

[목차 1](#_Toc11262101)

[1. 기획 2](#_Toc11262102)

[1.1. 개요 2](#_Toc11262103)

[1.2. 배경 2](#_Toc11262104)

[1.3. 게임 플레이 구성 2](#_Toc11262105)

[1.4. 과제 요구사항 충족 계획 3](#_Toc11262106)

[2. 실행 결과 4](#_Toc11262107)

[2.1.1. 구현된 항목 4](#_Toc11262108)

[3. 구현 상세 11](#_Toc11262109)

[3.1. 물리 - 등가속도 운동 11](#_Toc11262110)

[3.1.1. 적용 11](#_Toc11262111)

[3.1.2. 이론 11](#_Toc11262112)

[3.1.3. 구현 11](#_Toc11262113)

[3.2. 인공지능 – 유한상태기계 12](#_Toc11262114)

[3.2.1. 적용 12](#_Toc11262115)

[3.2.2. 구현 12](#_Toc11262116)

[3.3. 소리 13](#_Toc11262117)

[3.3.1. 구현 13](#_Toc11262118)

[3.4. 충돌 검사 및 거리 측정 15](#_Toc11262119)

[3.4.1. 적용 15](#_Toc11262120)

[3.4.2. 구현 15](#_Toc11262121)

[3.5. 회전변환 15](#_Toc11262122)

[3.5.1. 적용 15](#_Toc11262123)

[3.5.2. 이론 16](#_Toc11262124)

[3.5.3. 구현 16](#_Toc11262125)

[4. 참고 자료 17](#_Toc11262126)

# 기획

## 개요

마우스를 이용한 슈팅 게임

## 배경

플레이어는 우주를 돌아다니며 심심풀이로 블랙홀 주위를 공전하는 우주 암석을 미사일로 맞추어, 블랙홀 안으로 밀어 넣는다

## 게임 플레이 구성

블랙홀

발사대

미사일

혜성

마우스 포인터

미사일 경로

우주 암석과 우주 갈매기

운석 경로

배경 화면은 우주를 연상케 하는 비트맵 이미지.

마우스 포인터는 OS의 마우스 포인터를 그대로 사용.

발사 전 미사일의 방향은 마우스 포인터를 가리킴.

마우스 클릭 시(up event) 미사일이 발사되며, 경로는 클릭했던 위치를 향하는 반직선이다.

일정 시간이 지난 뒤 미사일이 재생성된다.

우주 갈매기는 중립 NPC로, 사용자의 미사일 발사 성과에 따라 반응한다.

## 과제 요구사항 충족 계획

효과음(2종류 이상, 4개 이상의 wav clip):

미사일 발사 시, 미사일 충돌 시, 미사일 발사 실패 시, 미사일 준비 완료 시

토글 키로 음소거: F2 키에 할당

볼륨 조절: F3과 F4 키에 할당

물리 - 등가속도: 미사일 비행에 적용

물리 – 충돌: 미사일이 다른 물체에 근접했을 때

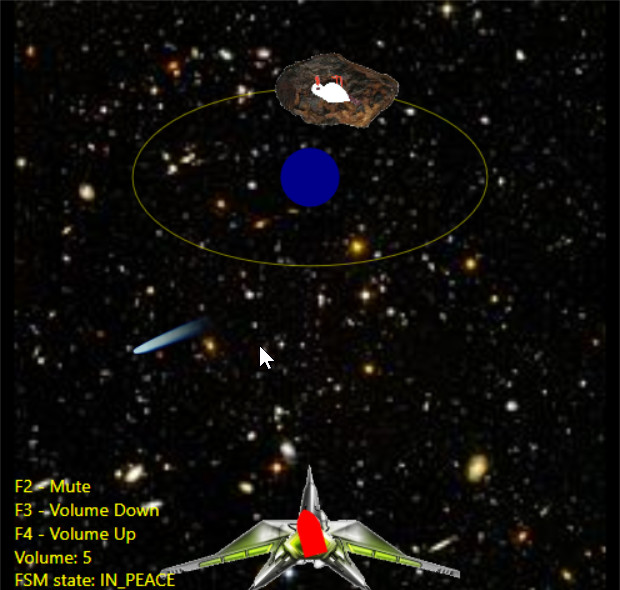
인공지능 – 유한상태기계: 우주 갈매기에 적용

# 실행 결과

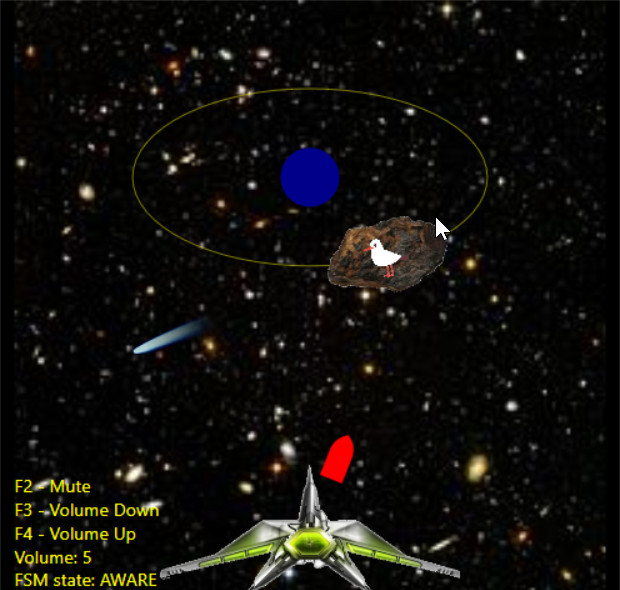
### 구현된 항목

#### 지난 번 과제에서 구현되지 않았던 항목

미사일의 방향을 마우스 포인터를 바라보는 방향으로 바꾸기 위한 회전변환 적용



미사일이 클릭한 방향으로 특정 시간만큼 비행



#### 과제 요구사항

효과음(2종류 이상, 4개 이상의 wav clip):

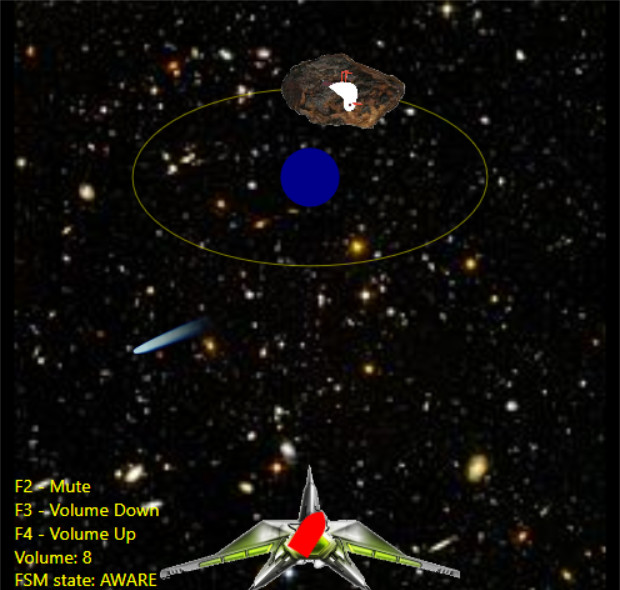
미사일 충돌 시, 미사일 발사 시, 미사일 발사 실패 시, 미사일 준비 완료 시

<화면 생략>

토글 키로 음소거: F2 키에 할당, 소리가 나지 않고 화면 하단 설명에 “(MUTE)” 라고 표시된다.



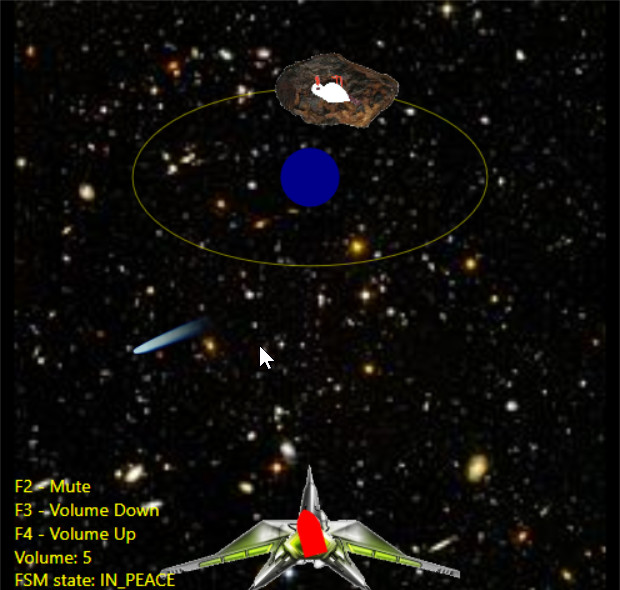
볼륨 조절: F3과 F4 키에 할당, 소리의 크기가 변하고 화면 하단 설명에 현재 크기가 표시된다.

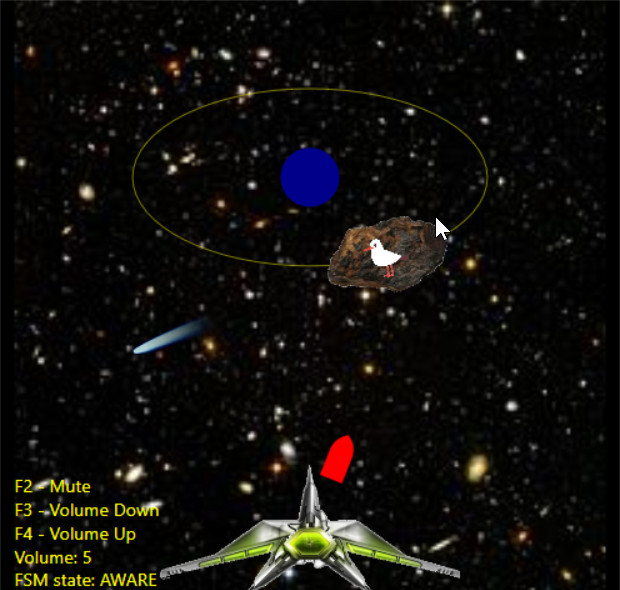


물리 - 등가속도: 미사일 비행 시 가속

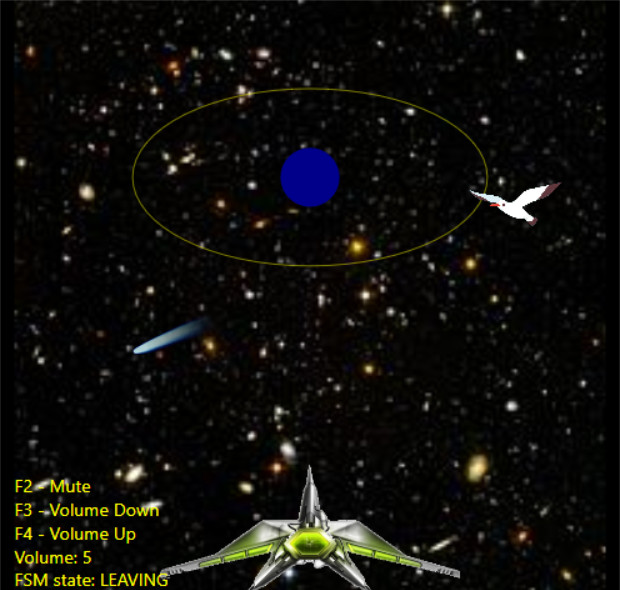
<화면 생략>

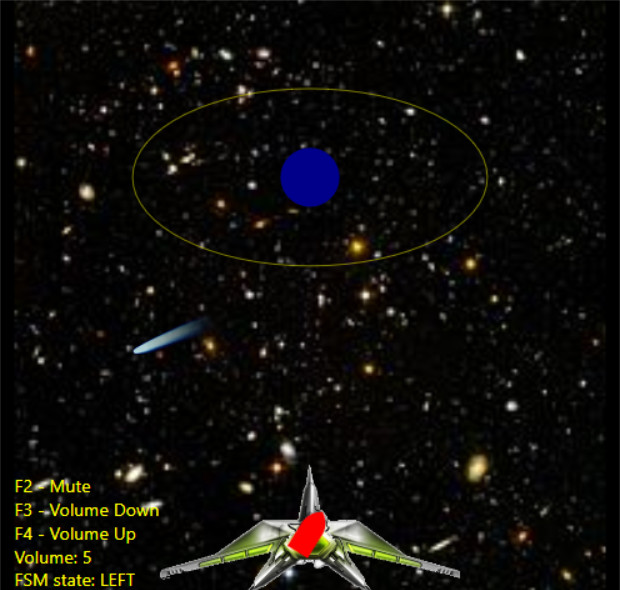
물리 – 충돌, 인공지능 - FSM: 미사일이 발사되었을 때, 근접했을 때 갈매기의 이미지가 바뀌고 미사일이 다른 물체에 최근접했을 때, 효과음이 나고 미사일과 우주 암석이 사라지고 갈매기가 화면의 오른쪽 너머로 천천히 이동한다. 갈매기 FSM 상태는 화면 하단에 표시된다.











# 구현 상세

## 물리 - 등가속도 운동

### 적용

미사일 비행 시 가속

### 이론

다음 공식 유도는 속도 0에서 등가속도 운동을 시작하여, PathGeometry 생성 시 지정한 Path의 끝까지 가는데 걸리는 시간이 duration parameter와 같게 하기 위해 한 것이다.

다음과 같은 등가속도 운동 공식에서,

(는 이동거리, 는 가속도 상수, 는 시간)

를 이동거리 에 도착하는 시간이라고 가정하면,

등가속도 운동이므로 는 상수이다.

따라서 위 공식을 다음과 같이 바꿔 쓸 수 있다.

### 구현

위 공식 유도를 통해, 초기 속도는 0이고, 가속도를 따로 지정하지 않고 PathGeometry에서 지정한 길이만큼 정확한 시간(duration)에 맞추어 움직인다. 최초 위치가 this->m\_Start로 정해지는 것 외에는 위 공식에 대입된다. (S = this->m\_End – this->m\_Start)

다음 코드는 파일 Animation.h에 있다.

template <typename T>

class AnimationAccel : public Animation<T> {

public:

AnimationAccel(T start = 0, T end = 0, T duration = 0)

: Animation<T>(start, end, duration) { }

protected:

virtual T ComputeValue(T time) {

return

this->m\_Start +

(this->m\_End - this->m\_Start) \* (time / this->m\_Duration) \* (time / this->m\_Duration);

}

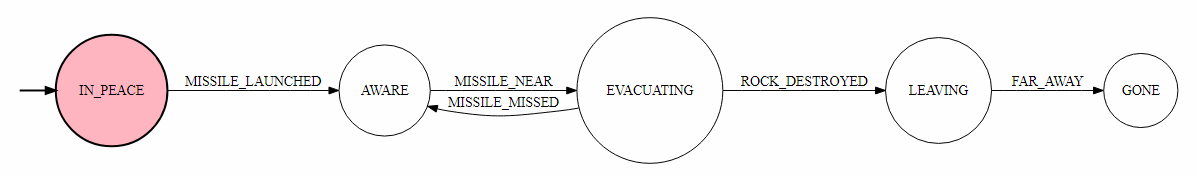
};

## 인공지능 – 유한상태기계

### 적용

중립 NPC인 우주 갈매기에 적용

### 구현



위 도식은 <http://ivanzuzak.info/noam/webapps/fsm_simulator/>에서 다음 문서로 만들 수 있다.

#states

IN\_PEACE

AWARE

EVACUATING

LEAVING

GONE

#initial

IN\_PEACE

#accepting

#alphabet

MISSILE\_LAUNCHED

MISSILE\_NEAR

MISSILE\_MISSED

ROCK\_DESTROYED

FAR\_AWAY

#transitions

IN\_PEACE:MISSILE\_LAUNCHED>AWARE

AWARE:MISSILE\_NEAR>EVACUATING

EVACUATING:MISSILE\_MISSED>AWARE

EVACUATING:ROCK\_DESTROYED>LEAVING

LEAVING:FAR\_AWAY>GONE

## 소리

### 구현

#### 믹싱

XAudio2를 사용하여 구현하였다.

다음 메서드의 정의에서는 CWaveFile을 이용하여 wav 파일을 로드하고 소스 보이스를 생성한다.

HRESULT DemoApp::LoadWave(LPWSTR filename, IXAudio2SourceVoice\*\* ppsv, XAUDIO2\_BUFFER\* buf);

소리 크기 조절은 DemoApp::WndProc에서 WM\_KEYDOWN 메시지를 받았을 때 한다.

다음 메서드는 XAudio2 마스터링 보이스의 소리를 logarithmic 하게 조절하기 위해 정의/사용하였다.

float DemoApp::GetVolumeLinear() const {

if (m\_IsMuted || m\_VolumeLog == 0) {

return 0.0F;

} else {

return 2.0F / pow(sqrt(2.0F), 10 - m\_VolumeLog);

}

}

다음과 같이 사용한다.

m\_pMasterVoice->SetVolume(pDemoApp->GetVolumeLinear());

#### 효과음

미사일 충돌 시, 미사일 발사 시, 미사일 발사 실패 시, 미사일 준비 완료 시에 효과음을 낸다.

미사일 충돌: DemoApp::OnRender에서 미사일이 우주 암석 영역에 도달 시

if (m\_GameState == GameState::PLAYING) {

const FLOAT d = Dist(rockCoord, missileCoord);

if (d <= 100.0F) {

// The missile hit the rock.

m\_MissileState = MissileState::HIT;

m\_GameState = GameState::SUCCEESS;

IssueSeagullEvent(SeagullEvent::ROCK\_DESTROYED);

m\_pHitVoice->Stop();

m\_pHitVoice->FlushSourceBuffers();

m\_pHitVoice->SubmitSourceBuffer(&m\_HitBuffer);

m\_pHitVoice->Start();

} else if (d <= 250.0F) {

// The missile come close to the rock.

IssueSeagullEvent(SeagullEvent::MISSILE\_NEAR);

}

}

미사일 발사: DemoApp::WndProc의 WM\_LBUTTONUP 이벤트 발생 시

미사일 발사 실패: DemoApp::WndProc의 WM\_LBUTTONUP 이벤트 발생 시, 여러 번 누를 경우 재생하던 소리를 끊고(Stop()) 소스의 처음부터 1번만 들리게 함(FlushSourceBuffers() 이후)

case WM\_LBUTTONUP :

if (pDemoApp->m\_IsMissileLaunched) {

pDemoApp->m\_pMissileNotReadyVoice->Stop();

pDemoApp->m\_pMissileNotReadyVoice->FlushSourceBuffers();

pDemoApp->m\_pMissileNotReadyVoice->Start();

pDemoApp->m\_pMissileNotReadyVoice->SubmitSourceBuffer(&pDemoApp->m\_MissileNotReadyBuffer);

} else {

pDemoApp->m\_IsMissileLaunched = true;

pDemoApp->m\_LButtonUp.x = LOWORD(lParam);

pDemoApp->m\_LButtonUp.y = HIWORD(lParam);

pDemoApp->m\_pMissileLaunchVoice->SubmitSourceBuffer(&pDemoApp->m\_MissileLaunchBuffer);

}

break;

미사일 준비 완료: DemoApp::OnRender에서 미사일 애니메이션이 종료되었을 때

if (missileAnimationTime >= m\_MissileAnimation.GetDuration()) {

// 미사일 애니메이션의 끝에 도달하면 발사되기 전의 우주선에 장착된 모습으로 되돌림.

missileAnimationTime = 0.0F;

m\_IsMissileLaunched = false;

m\_pMissileReadyVoice->SubmitSourceBuffer(&m\_MissileReadyBuffer);

} else if (m\_IsMissileLaunched) {

// 미사일이 날아가고 있으면 시간 추가.

missileAnimationTime += elapsedTime;

}

## 충돌 검사 및 거리 측정

### 적용

미사일이 우주 암석에 근접했을 때

### 구현

거리 측정 함수의 정의는 다음과 같다.

FLOAT Dist(D2D1\_POINT\_2F p1, D2D1\_POINT\_2F p2) {

return sqrt((p1.x - p2.x) \* (p1.x - p2.x) + (p1.y - p2.y) \* (p1.y - p2.y));

}

위 함수를 이용해 DemoApp::OnRender에서 일정 거리 이내로 미사일과 우주 암석이 가까워졌을 때 적절한 이벤트를 발생시킨다.

if (m\_GameState == GameState::PLAYING) {

const FLOAT d = Dist(rockCoord, missileCoord);

if (d <= 100.0F) {

// The missile hit the rock.

m\_MissileState = MissileState::HIT;

m\_GameState = GameState::SUCCEESS;

IssueSeagullEvent(SeagullEvent::ROCK\_DESTROYED);

m\_pHitVoice->Stop();

m\_pHitVoice->FlushSourceBuffers();

m\_pHitVoice->SubmitSourceBuffer(&m\_HitBuffer);

m\_pHitVoice->Start();

} else if (d <= 250.0F) {

// The missile come close to the rock.

IssueSeagullEvent(SeagullEvent::MISSILE\_NEAR);

}

}

## 회전변환

### 적용

마우스 포인터 방향을 가리키는 미사일 준비 상태

클릭한 방향으로 비행하는 미사일

### 이론

발사대

미사일

θ

마우스 포인터

변환되기 전 미사일 애니메이션의 기본 경로가 주황색 점선으로 표시되어 있다. 이 선을 기준으로 θ 만큼 회전 변환을 하면 된다.

변환되기 전 미사일 애니메이션의 기본 경로의 방향은 화면 좌표계에서 -y 방향이다. 따라서,

### 구현

translatedCoord는 우주선을 원점으로 하는 좌표이다. 미사일이 발사됐는지 아닌지에 따라 마우스 포인터의 좌표를 쓸지, 클릭한 좌표를 쓸지 결정한다.

다음 코드는 DemoApp::OnRender에 있다.

const D2D1\_POINT\_2F& translatedCoord =

m\_IsMissileLaunched ? translatedLButtonUp : translatedMouseMove;

const float r =

std::sqrt(

translatedCoord.x \* translatedCoord.x +

translatedCoord.y \* translatedCoord.y

);

const float s = translatedCoord.x / r;

const float c = -translatedCoord.y / r;

const D2D1\_MATRIX\_3X2\_F rotation = D2D1::Matrix3x2F(

c, s,

-s, c,

0.0F, 0.0F

);

# 참고 자료

박종승 교수님, 수업 자료, 실습 자료