게임 프로그래밍

HW3 보고서

박종승 교수님

컴퓨터 공학부

201401414 곽용준

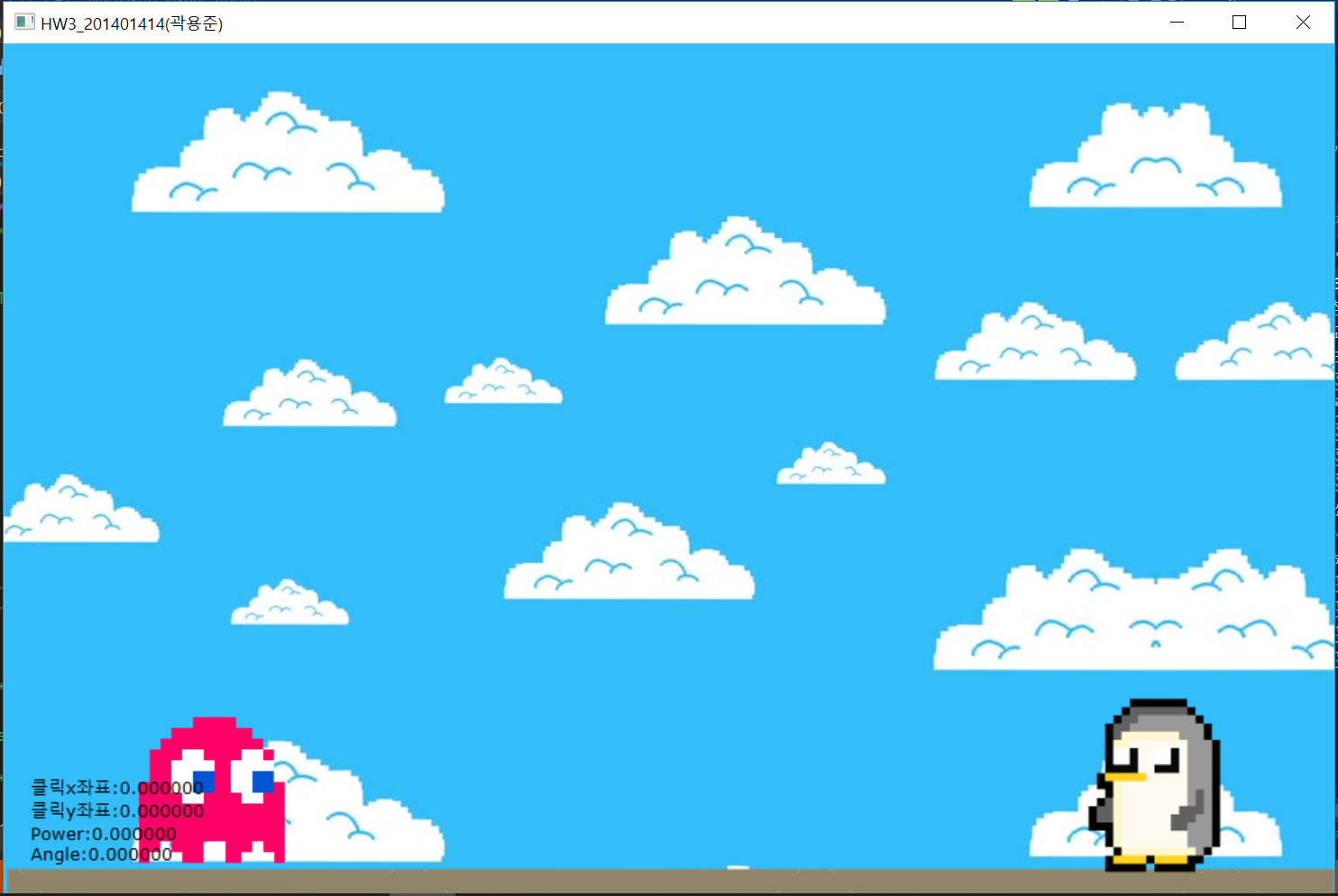
∙ 키 조작

초기 화면은 좌측에는 플레이어 케릭터, 우측에는 적 케릭터가 보인다. 플레이어는 클릭을 통하여 미사일을 발사 할 수 있다.

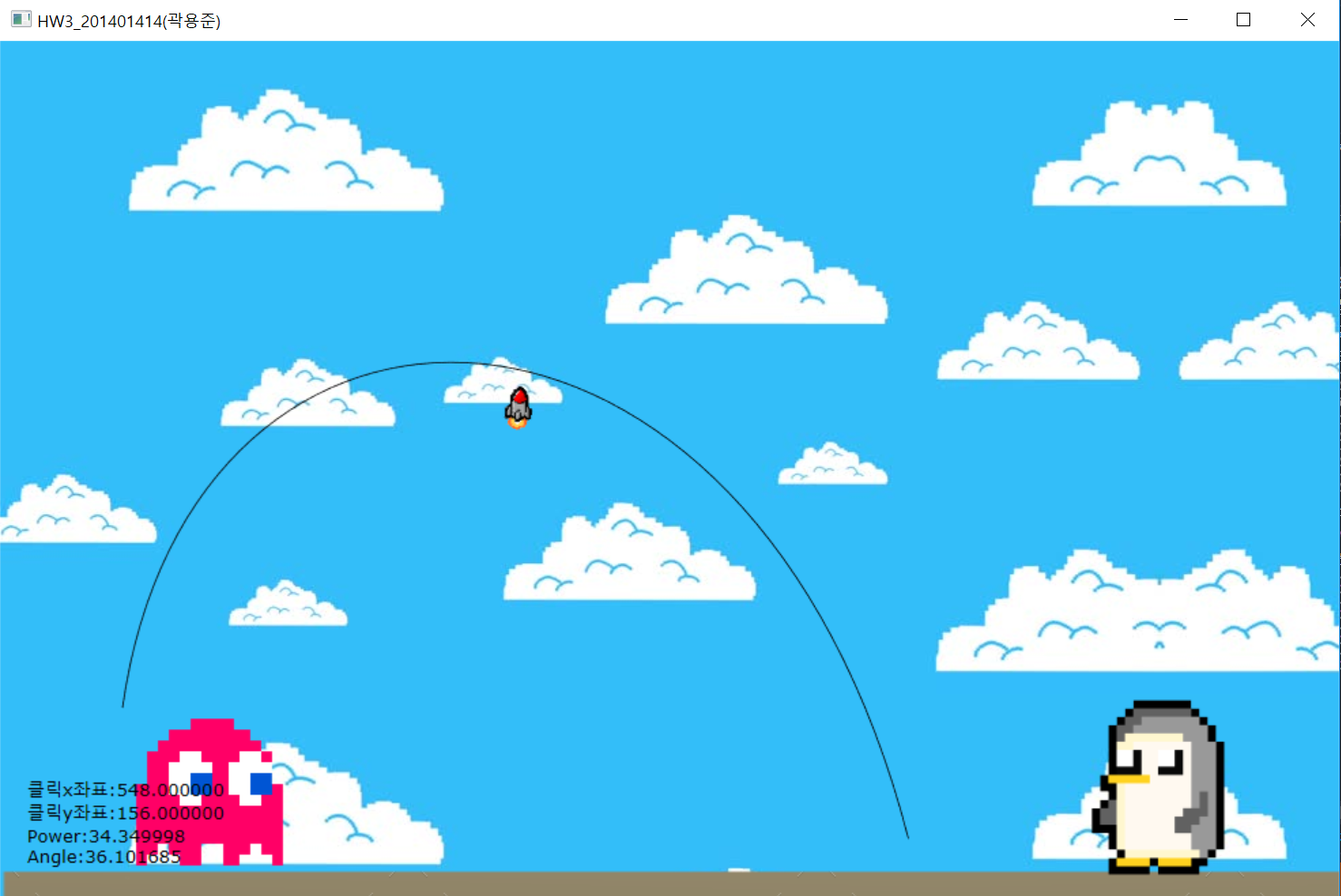
플레이어가 클릭한 위치와 플레이어 케릭터 간의 각도와 누른 시간을 이용하여 발사각과 힘을 계산하여 미사일을 발사한다.

∙ 실행 결과

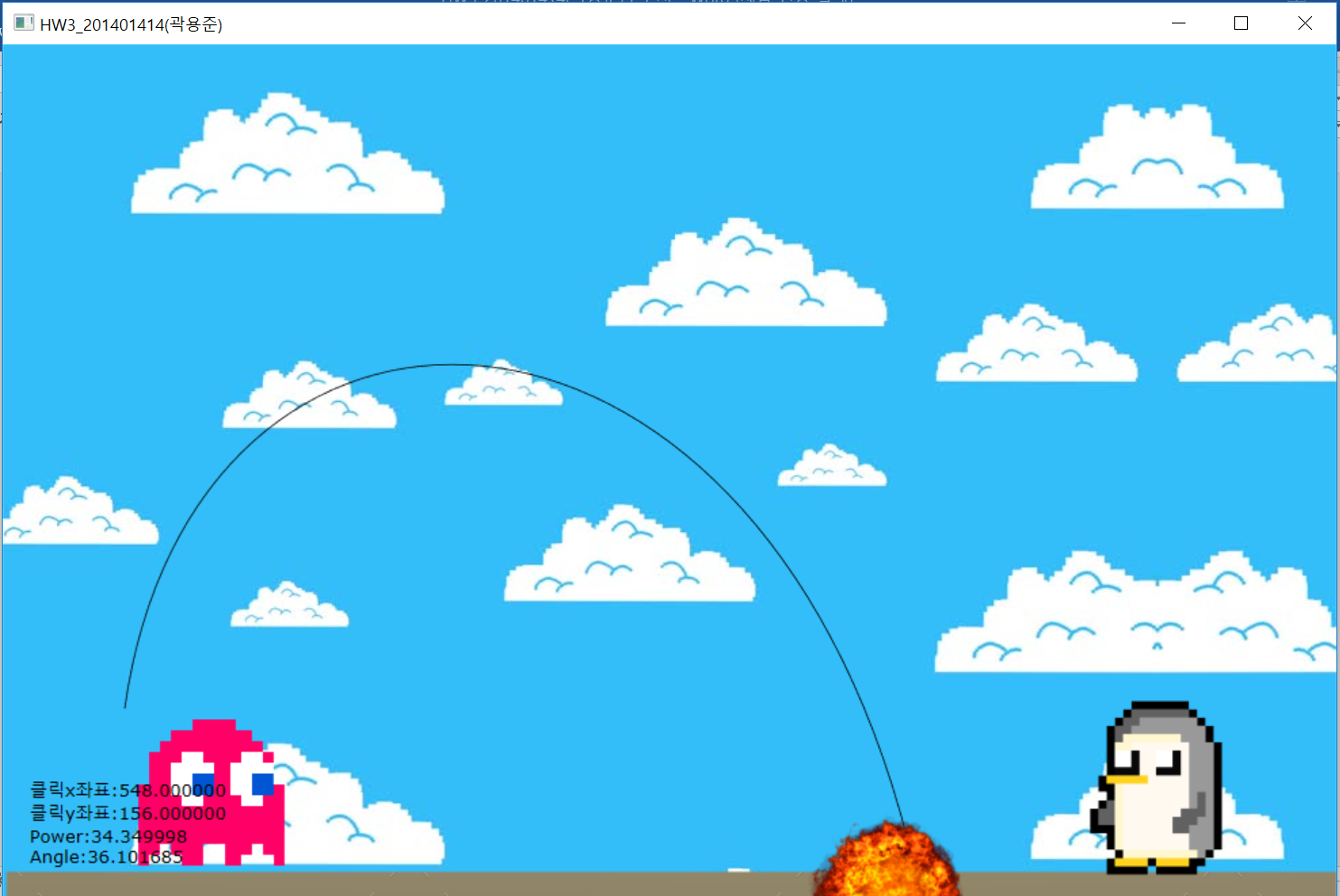
초기화면



클릭시 미사일 발사



바닥에 닿으면 폭발



∙ 주요 코드

**경로 기하 생성**

point1.x = 150;

point1.y = 50;

point2.x = 650;

point2.y = 50;

BezierEnd.x = p2\_location.x;

BezierEnd.y = p2\_location.y + 150;

pSink->BeginFigure(p1\_location, D2D1\_FIGURE\_BEGIN\_FILLED);

pSink->AddBezier(D2D1::BezierSegment(point1, point2, BezierEnd));

pSink->EndFigure(D2D1\_FIGURE\_END\_OPEN);

hr = pSink->Close();

장치 독립적 자원인 경로기하는 다음과 같이 미리 생성해 둔다.

**미사일 생성 및 경로 기하 조정**

//미사일 궤적 그리기

if (isShow) {

Angle = atan2(p1\_location.y - click\_p.y, click\_p.x - p1\_location.x) \* 180 / PI;

FLOAT middle = 0.0f;

Time = Power \* sin(getRadian(Angle));

MissileEnd.x = (Power \* cos(getRadian(Angle)) \* Time) + p1\_location.x;

MissileEnd.y = rtSize.height / 2 + 230;

if (Angle == 0) {

y\_rate = 0;

}

else {

y\_rate = Angle / 45;

}

x\_rate = MissileEnd.x / p2\_location.x;

D2D1::Matrix3x2F missileScale = D2D1::Matrix3x2F::Scale(D2D1::Size(x\_rate, y\_rate), p1\_location);

m\_pRT->SetTransform(missileScale);

m\_pRT->DrawGeometry(m\_pPathGeometry, m\_pBlackBrush);

static float anim\_time = 0.0f;

static float elapsedTime = 0.0;

float length = m\_Animation.GetValue(anim\_time);

m\_pPathGeometry->ComputePointAtLength(length, NULL, &point, &tangent);

//미사일 크기 구하기

D2D1\_SIZE\_F missileSize = m\_pMissileBitmap->GetSize();

D2D1\_RECT\_F rcMissileRect = D2D1::RectF(0, 0, missileSize.width, missileSize.height);

// 미사일의 방향을 조절하여 이동 동선을 따라가는 방향이 되도록 함.

triangleMatrix = D2D1::Matrix3x2F(

tangent.x,tangent.y,

-tangent.y,tangent.x,

point.x, point.y);

//m\_pRT->SetTransform(triangleMatrix \* scale \* translation);

m\_pRT->SetTransform(triangleMatrix \* missileScale);

// 미사일 그리기

m\_pRT->FillRectangle(rcMissileRect, m\_pMissileBitmapBrush);

if (isShoot) {

anim\_time = 0;

isShoot = false;

}

클릭한 시간을 계산하여 X비율을, 클릭 위치와 플레이어의 각도를 이용하여 Y비율을 계산하여 적절히 경로 기하를 확대 및 축소한다.

**애니메이션**

// 애니메이션의 끝에 도달하면 폭발이미지 생성

if (anim\_time >= m\_Animation.GetDuration())

{

D2D1\_SIZE\_F explosionSize = m\_pExplosionBitmap->GetSize();

D2D1\_RECT\_F rcExplosionRect = D2D1::RectF(0, 0, explosionSize.width, explosionSize.height);

D2D1\_MATRIX\_3X2\_F explosiontran = D2D1::Matrix3x2F::Translation(MissileEnd.x - explosionSize.width/2,MissileEnd.y);

m\_pRT->SetTransform(explosiontran);

m\_pRT->SetAntialiasMode(D2D1\_ANTIALIAS\_MODE\_ALIASED);

m\_pRT->FillOpacityMask(

m\_pExplosionBitmap,

m\_pExplosionBitmapBrush,

D2D1\_OPACITY\_MASK\_CONTENT\_GRAPHICS,

&rcExplosionRect,

NULL

);

m\_pRT->SetAntialiasMode(D2D1\_ANTIALIAS\_MODE\_PER\_PRIMITIVE);

}

else

{

LARGE\_INTEGER CurrentTime;

QueryPerformanceCounter(&CurrentTime);

float elapsedTime = (float)((double)(CurrentTime.QuadPart - m\_nPrevTime.QuadPart) / (double)(m\_nFrequency.QuadPart));

m\_nPrevTime = CurrentTime;

anim\_time += elapsedTime;

}

애니메이션의 시간을 계산하여 끝에 도달하였을 경우 경로기하의 끝 부분에 폭발하는 이미지를 그려 넣는다.