<공튀기기 게임>

1.주제설명: 라켓이있고 공이있다 공이 처음에 위에서 떨어진다 라켓이 공을치면 공은 날라가 벽에부딛힌다.부딛힌 공은 다시 반사각방향으로 나아간다. 라켓이 떨이지는 공을 받아치면 ‘아싸 ’부분에 1씩 점수가 추가되고 놓치면 ‘아쉽 놓쳤네’ 부분에 1이 추가가 된다 게임은 1분후에 종료가된다. 메뉴 상단에 보면 난이도 부분이 있다. 난이도 부분을 내가 조절할 수 있다 난이도가 어려울수록 공의 속도가 빨라진다

2. 소스코드(CBitnaGameView)

// BitnaGameView.cpp : CBitnaGameView 클래스의 구현

//

#include "stdafx.h"

// SHARED\_HANDLERS는 미리 보기, 축소판 그림 및 검색 필터 처리기를 구현하는 ATL 프로젝트에서 정의할 수 있으며

// 해당 프로젝트와 문서 코드를 공유하도록 해 줍니다.

#ifndef SHARED\_HANDLERS

#include "BitnaGame.h"

#endif

#include "BitnaGameDoc.h"

#include "BitnaGameView.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// CBitnaGameView

IMPLEMENT\_DYNCREATE(CBitnaGameView, CView)

BEGIN\_MESSAGE\_MAP(CBitnaGameView, CView)

// 표준 인쇄 명령입니다.

ON\_COMMAND(ID\_FILE\_PRINT, &CView::OnFilePrint)

ON\_COMMAND(ID\_FILE\_PRINT\_DIRECT, &CView::OnFilePrint)

ON\_COMMAND(ID\_FILE\_PRINT\_PREVIEW, &CView::OnFilePrintPreview)

ON\_WM\_KEYDOWN()

ON\_WM\_LBUTTONDOWN()

ON\_WM\_TIMER()

ON\_COMMAND(ID\_32771, &CBitnaGameView::OnEasy)

ON\_COMMAND(ID\_32772, &CBitnaGameView::OnMidum)

ON\_COMMAND(ID\_32773, &CBitnaGameView::Onhard)

END\_MESSAGE\_MAP()

// CBitnaGameView 생성/소멸

CBitnaGameView::CBitnaGameView()

: ballStartPoint(0)

{

// TODO: 여기에 생성 코드를 추가합니다.

stepsOf.cx = 25; //공의 시작좌표를 잡아준다

stepsOf.cy = -25;

racket.SetRect(0, 0, 100, 10);// 라켓의 rect를 설정한다

isTimerSet = false;//타이머를 꺼둔다

close = false;//게임을 끝낼때 쓴다

}

CBitnaGameView::~CBitnaGameView()

{

}

BOOL CBitnaGameView::PreCreateWindow(CREATESTRUCT& cs)

{

// TODO: CREATESTRUCT cs를 수정하여 여기에서

// Window 클래스 또는 스타일을 수정합니다.

CBitmap heart;//wallpaper를 입혀줌

heart.LoadBitmapA(bbb);

CBrush brush(&heart);

cs.lpszClass = AfxRegisterWndClass(CS\_HREDRAW|CS\_VREDRAW|CS\_DBLCLKS,

::LoadCursor(NULL, IDC\_ARROW),

//reinterpret\_cast<HBRUSH>(COLOR\_WINDOW+1),

(HBRUSH)brush.Detach(),NULL);

return CView::PreCreateWindow(cs);

}

// CBitnaGameView 그리기

void CBitnaGameView::OnDraw(CDC\* pDC)

{

CBitnaGameDoc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

if (!pDoc)

return;

if (!close) //아직 게임이 시작 안됨

{

CBitmap h;//공 비트맵을 가져옴

h.LoadBitmapA(bbbb);

BITMAP info;

h.GetBitmap(&info);

CDC menDC;

menDC.CreateCompatibleDC(pDC);

menDC.SelectObject(&h);

pDC->SetMapMode (MM\_LOENGLISH);//맵모드 설정해주기

ballStartPoint = pDoc->getBallPosition();//시작점을 가져온다

ballEndPoint.x = ballStartPoint.x + info.bmWidth;

ballEndPoint.y = ballStartPoint.y - info.bmHeight;

pDC->BitBlt(ballStartPoint.x, ballStartPoint.y,info.bmWidth,info.bmHeight,&menDC,0,0,SRCCOPY);//공을 그려줌

CBrush racketBrush (RGB(0, 255, 0));

pDC->SelectObject(&racketBrush);

CSize racketSize(100,10); //라켓 크기 잡아줌

racketStartPoint = pDoc->getRacketPosition();//라켓 좌표가져옴

racketEndPoint.x = racketStartPoint.x + racketSize.cx;

racketEndPoint.y = racketStartPoint.y - racketSize.cy;

racket.SetRect(racketStartPoint.x, racketStartPoint.y, racketEndPoint.x, racketEndPoint.y);//라켓의 영역을 잡아주고

pDC->Rectangle(racketStartPoint.x, racketStartPoint.y, racketEndPoint.x, racketEndPoint.y);//라켓을 그려준다

CFont fontW;//놓친공과 잡은 공 표시해주는 부분

fontW.CreatePointFont(300,"궁서");

pDC->SelectObject(&fontW);

CRect missedScreen;//놓쳤을때 글씨 표시해줄 영역

CRect catchedScreen;//잡앗다 글씨 표시해줄 영역

missedScreen.SetRect(1000, 0, 1220, -40 );

catchedScreen.SetRect(20,0, 100, -40 );

pDC->DrawText(\_T("아쉽 놓쳤다"), missedScreen, DT\_RIGHT);

pDC->DrawText(\_T("아싸! 잡았다"), catchedScreen, DT\_LEFT);

CFont fontS;//스코어 숫자 표시하기위해

fontS.CreatePointFont(500,"궁서");

pDC->SelectObject(&fontS);

CRect screenNumMissed(1000,-50, 1220, -125 );//놓친것 스코어 표시

CRect screenNumCatched(18,-50, 100, -125 );//잡은것 스코어 표시

CRect screenCount(540,-15, 700, -100 );//시간초 표시

CString num;

num.Format(\_T("%d"), pDoc->getMissedScore());

pDC->DrawText(num, screenNumMissed, DT\_RIGHT);

num.Format(\_T("%d"), pDoc->getCatchedScore());

pDC->DrawText(num, screenNumCatched, DT\_LEFT);

num.Format(\_T("%d"), pDoc->getCount());

pDC->DrawText(num, screenCount, DT\_CENTER|DT\_VCENTER|DT\_SINGLELINE|DT\_NOCLIP);

}

// TODO: 여기에 원시 데이터에 대한 그리기 코드를 추가합니다.

}

// CBitnaGameView 인쇄

BOOL CBitnaGameView::OnPreparePrinting(CPrintInfo\* pInfo)

{

// 기본적인 준비

return DoPreparePrinting(pInfo);

}

void CBitnaGameView::OnBeginPrinting(CDC\* /\*pDC\*/, CPrintInfo\* /\*pInfo\*/)

{

// TODO: 인쇄하기 전에 추가 초기화 작업을 추가합니다.

}

void CBitnaGameView::OnEndPrinting(CDC\* /\*pDC\*/, CPrintInfo\* /\*pInfo\*/)

{

// TODO: 인쇄 후 정리 작업을 추가합니다.

}

// CBitnaGameView 진단

#ifdef \_DEBUG

void CBitnaGameView::AssertValid() const

{

CView::AssertValid();

}

void CBitnaGameView::Dump(CDumpContext& dc) const

{

CView::Dump(dc);

}

CBitnaGameDoc\* CBitnaGameView::GetDocument() const // 디버그되지 않은 버전은 인라인으로 지정됩니다.

{

ASSERT(m\_pDocument->IsKindOf(RUNTIME\_CLASS(CBitnaGameDoc)));

return (CBitnaGameDoc\*)m\_pDocument;

}

#endif //\_DEBUG

// CBitnaGameView 메시지 처리기

void CBitnaGameView::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)

{

// TODO: 여기에 메시지 처리기 코드를 추가 및/또는 기본값을 호출합니다.

CBitnaGameDoc\* pDoc = GetDocument();

CSize offset(40,40); //한번에 움직이는 양

if (nChar == VK\_LEFT)//좌우 움직이는 거 해주기

{

if (racketStartPoint.x >= 0)

{

racketStartPoint.x -= offset.cx;//왼쪽으로가면 시작값은 마이너스된다

racketStartPoint.y = -780;//와이값은 밑단값으로 고정이다

pDoc->setRacketPosition(racketStartPoint);//변화되는 좌표값을 계속 업데이트해준다

}

}

if (nChar == VK\_RIGHT)

{

if (racketEndPoint.x <= 1230) //0보다 작거다 1230보다 크면 움직이지 않는다

{

racketStartPoint.x += offset.cx;//오른쪽으로가면 시작값은 플러스된다

racketStartPoint.y = -780;

pDoc->setRacketPosition(racketStartPoint);

}

}

if(nChar==VK\_SPACE){//스페이스를 누르면 게임이 시작된다

CBitnaGameDoc\* pDoc = GetDocument();

if (!isTimerSet && !pDoc->getGameOver()) // 게임의 시작값을 true로 바꾸어준다

{

frequency = pDoc->getFrequency();//난의도 에따른 시간초값을 받아옴

SetTimer(1, frequency, NULL);//난의도 조절을 위해 frequency 로 받아온다 난이도에 따라 빠르기를 변화시킨다

SetTimer(2, 1000, NULL);//1초마다 시간초 count 값을 바꾸기 위해

isTimerSet = true;

}

}

CView::OnKeyDown(nChar, nRepCnt, nFlags);

}

void CBitnaGameView::OnTimer(UINT\_PTR nIDEvent)

{

// TODO: 여기에 메시지 처리기 코드를 추가 및/또는 기본값을 호출합니다.

CBitnaGameDoc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

if (!pDoc)

return;

if (nIDEvent == 1) // 타이머 1이라면

{

CClientDC dc (this);

dc.SetMapMode (MM\_LOENGLISH);

CRect rcClient;

GetClientRect(&rcClient);

dc.DPtoLP(&rcClient);

CPoint racketTop(0,-770);//라켓 두께와 위치 정해줌 두께는 10임

CPoint racketBottom(0, -780);

CPoint ballMidPoint(ballStartPoint.x + 25, ballStartPoint.y + 25);//공의 중간좌표

CPoint racketMidPoint(racketStartPoint.x + 50, racketStartPoint.y + 5);//라켓의 중간좌표

CSize xSteps(30,0);

if (ballStartPoint.x <= 0)//공이 화면밖으로 안나가게해준다

{

if (stepsOf.cx < 0) //만약 공의 엑스좌표가 -로내려가게되면 -로 계속 내려가면 공이 화면밖으로 나가버림

{

stepsOf.cx = -stepsOf.cx; //플러스로 바꾸어준다

}

}

if (ballEndPoint.x >= 1250) //공이 그려지는 끝좌표가 화면의 최대값을 못넘어가게 한다

//(시작좌표로 잡으면 공이화면밖으로 살짝 나갔다가 튕겨진다그래서 마지막 끝점으로 지정해줘야함 왜냐 끝점이 오른쪽면에닿으니까 )

{

if (stepsOf.cx > 0)

{

stepsOf.cx = -stepsOf.cx;

}

}

if (ballStartPoint.y >= 0)//공이 화면 제일 상단으로 나가지 않게하기 위해 잡아준다 x와 같은원리

{

if (stepsOf.cy > 0)

{

stepsOf.cy = -stepsOf.cy;

}

}

if (ballEndPoint.y <= -880)//공이 제일 하단으로 못나가게한다

{

if (stepsOf.cy < 0)

{

stepsOf.cy = -stepsOf.cy;

pDoc->setMissedScore(); //만약 공의 y값이 화면의 제일 하단의 좌표와 일치하면 놓친걸로 간주

}

}

if ((ballEndPoint.x >= racketStartPoint.x) && (ballStartPoint.x <= racketEndPoint.x) && (ballStartPoint.y <= racketTop.y))

{ //만약 공의 끝점좌표와 라켓으 시작좌표와 같거나 (공이 라켓 왼쪽면에 부딛혔을때), 공의 시작좌표와 라켓의 끝점이 같거나(오른쪽면에 부딛힘)

//공의 끝점이 라켓의 상단 좌표와 같다면 (좌, 우, 위 부딛혔을경우를 해줌)

if (stepsOf.cy < 0)

{

pDoc->setCatchedScore(); // 라켓가 부딛히면 공을 잡은걸로 인정

stepsOf.cy = -stepsOf.cy; //튕기는 효과를 위해 떨어지는 걸 위로 올려줌

stepsOf.cx = xSteps.cx \* ( (ballMidPoint.x - racketMidPoint.x) / (float)40); //다시 튕기는것에대한 x값을 위한 알고리즘

}

}

ballStartPoint.x += stepsOf.cx;//리바운드 댈떄 시작좌표 다시 잡아줌

ballStartPoint.y += stepsOf.cy;

pDoc->setBallPosition(ballStartPoint);//시작좌표료 다시 세팅

}

else if(nIDEvent == 2) // SERVICE COUNTDOWN TIMER

{

if (pDoc->getCount() > 0)//타이머시간이 계속 남아 있으면

{

pDoc->setCount(); //시간값을 마이너스해줌

}

else // 시간이 끝나면

{

KillTimer(2); //타이머들 종료시킴

KillTimer(1);

isTimerSet = false;

pDoc->setGameOver(); // 게임오버시킨다

}

}

CView::OnTimer(nIDEvent);

}

void CBitnaGameView::OnEasy()

{

// TODO: 여기에 명령 처리기 코드를 추가합니다.

CBitnaGameDoc\* pDoc = GetDocument();

pDoc->setFrequency(130);//게임 난의도 조절임 130ms마다 공이 한번씩 떨어짐

frequency=pDoc->getFrequency();

if (isTimerSet)

{

SetTimer(1, frequency, NULL);

}

}

void CBitnaGameView::OnMidum()

{

// TODO: 여기에 명령 처리기 코드를 추가합니다

CBitnaGameDoc\* pDoc = GetDocument();

pDoc->setFrequency(90);//타이머 90ms마다 공이 한번씩 떨어짐

frequency=pDoc->getFrequency();

if (isTimerSet)

{

SetTimer(1, frequency, NULL);

}

}

void CBitnaGameView::Onhard()

{

// TODO: 여기에 명령 처리기 코드를 추가합니다.

CBitnaGameDoc\* pDoc = GetDocument();

pDoc->setFrequency(50);//50ms 마다 공이 한번씩 떨어짐

frequency=pDoc->getFrequency();

if (isTimerSet)

{

SetTimer(1, frequency, NULL);

}

}

2. 소스코드 (CBitnaGameDoc)

BOOL CBitnaGameDoc::OnNewDocument()

{

if (!CDocument::OnNewDocument())

return FALSE;

// TODO: 여기에 재초기화 코드를 추가합니다.

// SDI 문서는 이 문서를 다시 사용합니다.

gameOver = false;//모든값의 초기화

missedScore = 0;

catchedScore = 0;

count = 60;

ballPosition.x = 190;

ballPosition.y = -200;

racketPosition.x = 500;

racketPosition.y = -780;

timerFrequency = 130;

return TRUE;

}

//초기화부분

void CBitnaGameDoc::setMissedScore(void)

{

missedScore++;//공을 놓치면 ++

}

int CBitnaGameDoc::getMissedScore()

{

return missedScore;//현재 놓친공 스코어 보내줌

}

void CBitnaGameDoc::setCatchedScore()

{

catchedScore++;//공을 잡음

}

int CBitnaGameDoc::getCatchedScore()

{

return catchedScore;//잡은공 정보 보내줌

}

void CBitnaGameDoc::setCount()

{

count--;//시간값을 계속 마이너스해줌

UpdateAllViews(NULL);//Invalidate같은 존재

}

int CBitnaGameDoc::getCount()

{

return count;//카운트값 보내줌

}

void CBitnaGameDoc::setBallPosition(CPoint newPos)//공의 위치가 바뀔떄마다 좌표값 바꾸어줌

{

ballPosition = newPos;

UpdateAllViews(NULL);

}

CPoint CBitnaGameDoc::getBallPosition()

{

return ballPosition;//공의 위치값을 준다

}

void CBitnaGameDoc::setRacketPosition(CPoint newPos)

{

racketPosition = newPos;//키보드로 조절하면서 계속 바뀌는 시작값을 새로 계속 지정해준다

UpdateAllViews(NULL);//CChildview 에서의 invalidate의 역할이다

}

CPoint CBitnaGameDoc::getRacketPosition()

{

return racketPosition;//라켓 위치정보 가져옴

}

void CBitnaGameDoc::setGameOver()//게임종료시

{

gameOver = true;

}

bool CBitnaGameDoc::getGameOver()

{

return gameOver;

}

int CBitnaGameDoc::getFrequency(void)

{

return timerFrequency;//게임 난의도 설정해주기

}

void CBitnaGameDoc::setFrequency(int f)

{

timerFrequency = f;

}

//기타 다른함수들

결과화면



시작화면



게임중간에 메뉴를 클릭함



게임이 끝나면 정지한다.

1. 구현방법
2. 어떻게 라켓은 움직이는가

OnDraw 에서 라켓을 그려준다

CSize racketSize(100,10); //라켓 크기 잡아줌

racketStartPoint = pDoc->getRacketPosition();//라켓 좌표가져옴

racketEndPoint.x = racketStartPoint.x + racketSize.cx;

racketEndPoint.y = racketStartPoint.y - racketSize.cy;

racket.SetRect(racketStartPoint.x, racketStartPoint.y, racketEndPoint.x, racketEndPoint.y);//라켓의 영역을 잡아주고

pDC->Rectangle(racketStartPoint.x, racketStartPoint.y, racketEndPoint.x, racketEndPoint.y);//라켓을 그려준다

먼저 라켓의 무효영역을 잡아준다 그리고 라켓의 좌표는 시작점과 끝점으로 결정된다. 시작점은 기존에 초기화된 값을 가져오고 끝점은 시작점에서 라켓 무효영역 사이즈만큼이다 여기서 –racketSize.cy 인 이유는 맵모드를 MM\_LOENGLISH 로 설정해두었기 때문이다 아래로 내려갈수록 –값이 커진다.

라켓을 움직이기 위해서는 좌우 키보드 값을 이용한다.

CSize offset(40,40); //한번에 움직이는 양

if (nChar == VK\_LEFT)//좌우 움직이는 거 해주기

{

if (racketStartPoint.x >= 0)

{

racketStartPoint.x -= offset.cx;//왼쪽으로가면 시작값은 마이너스된다

racketStartPoint.y = -780;//와이값은 밑단값으로 고정이다

pDoc->setRacketPosition(racketStartPoint);//변화되는 좌표값을 계속 업데이트해준다

}

}

if (nChar == VK\_RIGHT)

{

if (racketEndPoint.x <= 1230) //0보다 작거다 1230보다 크면 움직이지 않는다

{

racketStartPoint.x += offset.cx;//오른쪽으로가면 시작값은 플러스된다

racketStartPoint.y = -780;

pDoc->setRacketPosition(racketStartPoint);

}

0보다 크거나 1230보다 작을떄만 이동할 수 있다 이 값을 설정해주지 않으면 라켓이 밖으로 나가버린다 오른쪽 왼쪽 한번씩 클릭할 때 마다 offset 만큼 이동한다.

라켓은 위아래로 움직이지 않음으로 -780이라는 하단값에 고정이 된다. x값에만 변화를준다

변화가 일어날때마다 setRacketPosition(racketStartPoint); 함수를 이용해 라켓의 위치를 set해준다.

1. 공은 어떻게 움직이는가

먼저 OnDraw에서 공을 그려준다 공은 비트맵을 불러와 그려준다

ballStartPoint = pDoc->getBallPosition();//시작점을 가져온다

ballEndPoint.x = ballStartPoint.x + info.bmWidth;

ballEndPoint.y = ballStartPoint.y - info.bmHeight;

pDC->BitBlt(ballStartPoint.x, ballStartPoint.y,info.bmWidth,info.bmHeight,&menDC,0,0,SRCCOPY);//공을 그려줌

공의 시작점과 끝점을 잡는원리도 라켓과 동일하다. 여기서는 BitBlt로 공을 그려준다

먼저 공이 벽에 부딪혔을때의 이벤트처리이다

if (ballStartPoint.x <= 0)//공이 화면밖으로 안나가게해준다

{

if (stepsOf.cx < 0) //만약 공의 엑스좌표가 -로내려가게되면 -로 계속 내려가면 공이 화면밖으로 나가버림

{

stepsOf.cx = -stepsOf.cx; //플러스로 바꾸어준다

}

}

if (ballEndPoint.x >= 1250) //공이 그려지는 끝좌표가 화면의 최대값을 못넘어가게 한다

//(시작좌표로 잡으면 공이화면밖으로 살짝 나갔다가 튕겨진다그래서 마지막 끝점으로 지정해줘야함 왜냐 끝점이 오른쪽면에닿으니까 )

{

if (stepsOf.cx > 0)

{

stepsOf.cx = -stepsOf.cx;

}

}

if (ballStartPoint.y >= 0)//공이 화면 제일 상단으로 나가지 않게하기 위해 잡아준다 x와 같은원리

{

if (stepsOf.cy > 0)

{

stepsOf.cy = -stepsOf.cy;

}

}

if (ballEndPoint.y <= -880)//공이 제일 하단으로 못나가게한다

{

if (stepsOf.cy < 0)

{

stepsOf.cy = -stepsOf.cy;

pDoc->setMissedScore(); //만약 공의 y값이 화면의 제일 하단의 좌표와 일치하면 놓친걸로 간주

}

}

만약 공의 x 좌표가 0보다 작아질경우 그 값에 –를 해주어서 반대로 다시 점점 커지게 구현을 하였다

공이 x 좌표가 1250보다 커질경우 즉 오른쪽 화면보다 커질경우에도 –를 넣어 점점 작아지게 해준다.

y값도 똑같이 top 인 0 부터 button 값인 -880까지 범위로 두고 그 범위를 넘어가는 경우 –를 넣어 다시 작아지거나 커지게 해준다.

또 공이 하단값인 -880과 좌표가 일치하게 될 경우 공을 놓쳤다고 간주하고 misscatch의 값을 1을 추가해준다

공이 라켓에 맞은 경우는 다음과같다

if ((ballEndPoint.x >= racketStartPoint.x) && (ballStartPoint.x <= racketEndPoint.x) && (ballEndPoint.y <= racketTop.y))

{ //만약 공의 끝점좌표와 라켓으 시작좌표와 같거나 (공이 라켓 왼쪽면에 부딛혔을때), 공의 시작좌표와 라켓의 끝점이 같거나(오른쪽면에 부딛힘)

//공의 끝점이 라켓의 상단 좌표와 같다면 (좌, 우, 위 부딛혔을경우를 해줌)

if (stepsOf.cy < 0)

{

pDoc->setCatchedScore(); // 라켓가 부딛히면 공을 잡은걸로 인정

stepsOf.cy = -stepsOf.cy; //튕기는 효과를 위해 떨어지는 걸 위로 올려줌

stepsOf.cx = xSteps.cx \* ( (ballMidPoint.x - racketMidPoint.x) / (float)40 ); //다시 튕기는것에대한 x값을 위한 알고리즘

}

}

앞에서 설정해준 라켓의 위와 양 옆면의 값을 잡아주고 만약 공의 좌표가 라켓의 위 또는 양옆에 닿을경우에는 잡았다라고 간주를 하고 score를 1증가시킨다. 부딛힌 순간부터 y값을 점점 커지게 하기위해 –를 대입하고 x값의 경우는 튕기는 효과를 계산한 알고리즘을 쓴다 이 알고리즘은 제가 만든게아니라 인터넷에서 참고하여 만들었습니다.

1. 게임은 어떻게 시작이 되는가

게임은 스페이스바를 누르면 시작하는걸로 했습니다.

if(nChar==VK\_SPACE){//스페이스를 누르면 게임이 시작된다

CBitnaGameDoc\* pDoc = GetDocument();

if (!isTimerSet && !pDoc->getGameOver()) // 게임의 시작값을 true로 바꾸어준다

{

frequency = pDoc->getFrequency();//난이도 에따른 시간초값을 받아옴

SetTimer(1, frequency, NULL);//난이도 조절을 위해 frequency 로 받아온다 난이도에 따라 빠르기를 변화시킨다

SetTimer(2, 1000, NULL);//1초마다 시간초 count 값을 바꾸기 위해

isTimerSet = true;

}

스페이스바를 누르면 게임의 시작과 종료를 알리는 getGameOver()의 함수안에 있는 gameOver값을 가져온다 isTimerSet 함수는 타이머의 시작을 알린다 이들의 초기값은 false 이다 타이머는 두가지가 있다 고유값1을 가지고 있는 타이머는 난이도를 위한 것이다. 난이도값에따라 frequency 값이 달라지는데 어려울수록 값이 작아져 갱신이 빨리되어 공이 빠르게 떨어진다. 고유값 2를가지고 있는 타이머는 시간카운터를 위한것이다 60에서부터 일초마다 1씩떨어지고 0이되면 게임은 종료가된다 isTimerSet을 true로 바꾸면서 게임이 시작된다

1. 난이도는 어떻게 조절이 되는가

난이도는 공의 스피드에 따라 쉬움, 중간, 어려움으로 나누어진다 메뉴에 쉬움, 중간, 어려움을 만들어주고 각각에대한 이벤트 처리기를 만들어주었다.

void CBitnaGameView::OnEasy()

{

// TODO: 여기에 명령 처리기 코드를 추가합니다.

CBitnaGameDoc\* pDoc = GetDocument();

pDoc->setFrequency(130);//게임 난의도 조절임 130ms마다 공이 한번씩 떨어짐

frequency=pDoc->getFrequency();

if (isTimerSet)

{

SetTimer(1, frequency, NULL);

}

}

void CBitnaGameView::OnMidum()

{

// TODO: 여기에 명령 처리기 코드를 추가합니다

CBitnaGameDoc\* pDoc = GetDocument();

pDoc->setFrequency(90);//타이머 90ms마다 공이 한번씩 떨어짐

frequency=pDoc->getFrequency();

if (isTimerSet)

{

SetTimer(1, frequency, NULL);

}

}

void CBitnaGameView::Onhard()

{

// TODO: 여기에 명령 처리기 코드를 추가합니다.

CBitnaGameDoc\* pDoc = GetDocument();

pDoc->setFrequency(50);//50ms 마다 공이 한번씩 떨어짐

frequency=pDoc->getFrequency();

if (isTimerSet)

{

SetTimer(1, frequency, NULL);

}

}

난이도에 따라 frequency 의 값을 새로 세팅을 해준다 if(isTimeSet)을 해준이유는 이걸해줘야 게임도중에도 난이도 조절이 가능하다 저게 없으면 게임중에는 난이도 값이 바뀌지 않는다.

1. 1분후 종료는 어떻게 구현했는가

else if(nIDEvent == 2) // SERVICE COUNTDOWN TIMER

{

if (pDoc->getCount() > 0)//타이머시간이 계속 남아 있으면

{

pDoc->setCount(); //시간값을 마이너스해줌

}

else // 시간이 끝나면

{

KillTimer(2); //타이머들 종료시킴

KillTimer(1);

isTimerSet = false;

pDoc->setGameOver(); // 게임오버시킨다

}

}

getCount 는 count 값을 가져온다 초기에 count값을 0으로 초기화를 시켜주었다. setCount를 통해 1초마다 -1씩해준다 1초마다라는 설정은 이미 SetTimer(2,1,NULL)에서 해주었다. Count의 값이 0보다 작아짐녀 KillTimer를 통해 앞에서 등록된 두개의 타이머를 꺼준다. 또한 gameOver의 값도 false 로 바꾸어 줌으로써 종료가 완료된다.

1. 학습한 내용

이번에 새로 배운개념은 SetTimer와 그 메시지를 처리해주는 OnTimer 그리고 타이머를 종료시켜주는 KillTimer이다. SetTimer(ID 값, ms단위의 시간값, NULL)로 구성이 된다 ms단위의 시간값 마다 이 함수는 OnTimer메소드를 실행시킨다 이 메소드 안에서 ID값이 같은 곳을 찾아가 그 부분에 대한것만 실행을 한다. 따라서 OnTimer안에서는 여러기의 타이머에대한 구현을 해줄수 있다. 아이디 값만 다르면 된다. KillTimer는 말그대로 타이머를 종료시켜주는 메소드이다. 이타이머를 통해 1분동안 게임을 하는 것, 공이 시간에 따라 움직이는 것을 구현하였다.

그다음 배운개념은 UpdateAllViews(NULL); 이다 이는 CChildView에서 Invalidate와 같은 개념이다 처음에 함수들을 DOC에 넣어주고 실행을 시켯더니 화면이 움직이지 않아서 DOC파일로가 Invalidate를 넣어주었더니 에러가 났다 그래서 인터넷에 찾아보니 CChildDoc에서는 Invalidate 대신에 UpdateAllView를 써준다는 사실을 알았다.

1. 아직 해결되지 않은 부분

공이 화면의 양옆부분에 튀었을떄는 닿자마자 튀는데 왠지 모르겠지만 top에 공이 닿았을때는 위로 살짝 들어갔다가 튕기는 모습을 볼 수 있다.

if (ballStartPoint.y >= 0)//공이 화면 제일 상단으로 나가지 않게하기 위해 잡아준다 x와 같은원리

{

if (stepsOf.cy > 0)

{

stepsOf.cy = -stepsOf.cy;

}

}

이것이 문제의 부분이다 y의 값이 top의 값인 0보다 커지면 –로 값을 줄여주는 부분인데 왜 0보다 뒤로 더 들어가는 지 아직 해결이 되지 않았다.



비트맵이 거꿀로 출력이된다 그림은 똑바로되어있는데 게임 출력시 거꿀로 출력되는 이유도 찾아내지 못하였다.