2D 스프라이트를 이용한 볼링 게임 구현

김홍준, 김성지, 권순각

동의대학교 컴퓨터소프트웨어공학과

An Implementation of Game Using 2D Sprite

Hong-jun Kim, Sun-gji Kim, Soon-kak Kwon

Department of computer software Engineering, Dong-eui University

E-mail : kingkkkk@hanmail.net, lakeofknight@hotmail.com, skkwon@deu.ac.kr

요약

2D 스프라이트는 일반적으로 2D게임에서 애니메이션 효과를 주기위하여 사용하는 방법으로서 이를 이용하여 많은 게임이 제작되고 있다. 본 논문에서는 2D 스프라이트를 이용하여 볼링 게임을 구현하는 효과적인 방법을 제시한다. 업데이트 부분과 렌더링 부분으로 게임 구조를 분리함으로서 간단하게 구현할 수 있는 장점을 가진다.

키워드

2d 스프라이트, 게임 제작, 볼링게임

Ⅰ. 서 론

2D 스프라이트(sprite)는 2D게임제작을 위해서 필수적으로 사용되는 방법이다. 2D 스프라이트는 게임에서 캐릭터와 같은 움직이는 비트맵이미지를 의미한다.

스프라이트로 애니메이션을 구현하기 위해서는 비트맵이미지를 Frame으로 시간적으로 자르고 잘라진 이미지를 고속으로 회전시키고 출력시킴으로서 애니메이션 효과를 줄 수 있게 된다. 이러한 스프라이트된 애니메이션을 가지고 게임이 제작된다.

본 논문에서는 스프라이트가 게임에 어떻게 사용되고 구현되는지 살펴본다. 비트맵이미지를 가지고 스프라이트를 구현하는 방법과 DirectX를 사용하여 볼링 게임을 구현하는 방법을 제시한다. DirectX는 Microsoft windows 상에서 게임 제작에 필요한 SDK(software development kit)를 제공하여 게임 제작에 이용할 수 있다[1].

본 논문에서 구현된 볼링게임은 엔진부와 처리부로 두 개의 부분으로 나뉘어지며 엔진부에서는 물리적엔진 부분을 담당하고 처리부는 로직부분을 수행한다. 이렇게 분리함으로써 게임 구현이 간단해지는 장점을 갖는다. Ⅱ장에서는 기존의 2D 스프라이트 기법에 대하여 살펴보고, Ⅲ장에서는 스프라이트를 구현하는 방법을 살펴본다. 그리고 Ⅳ장에서는 2D 스프라이트를 갖고 볼링게임이 구현하는 방법을 설명한다. 마지막 Ⅴ장에서는 결론을 맺는다.

Ⅱ. 2D 스프라이트 기법

스프라이트란 화면 표시에서 반복 이용되는 도형의 무늬를 등록하고, 그 무늬에 다른 그림을 겹쳐서 합성시키는 기능으로 고속 표시가 가능하며 연속되는 그림을 매끄럽게 움직일 수 있다[2].

1) Tile

2D 스프라이트를 애니메이션 처리하기 위해서는 비트맵이미지는 연속적인 비트맵이미지를 사용하는데 이를 Tile이라고 한다. 일반적으로 하나의 비트맵이미지에 연속된 프레임으로 구성된다[3].

그림 1에서는 5개의 프레임으로 구성된 Tile의 예를 들었다.

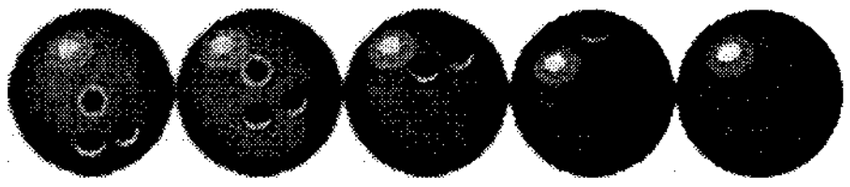

그림 2. 5개의 프레임으로 구성된 Tile

2) Cycle

2D sprite 애니메이션 Cycle은 게임에서 각 프레임의 애니메이션 중에 sprite를 이동시키는 과정이다. 스프라이트는 자주 이미지가 겹치게 되므로(배경과 캐릭터) 캐릭터와 같은 스프라이트는 자신이 가리고 있는 배경이미지를 먼저 출력 후 배경이미지 위에 캐릭터 이미지를 출력 하는 순서로 이동한다. 일반적인 스프라이트 애니메이션 Cycle은 다음과 같이 진행된다[3].

① 모든 스프라이트 삭제
② 각 스프라이트 위치 업데이트
③ 배경이미지 출력
④ 모든 스프라이트 출력
⑤ 위 ①~④ 반복

III. 2D 스프라이트 구현

스프라이트를 구현하기 위해서는 Tile형식으로 비트맵이미지를 제작해야 한다.

비트맵이미지의 파일이름, 프레임의 개수, 프레임의 폭, 높이 등을 가지고 있는 정보를 제작하여 게임코드와 리소스간의 직교성을 높여준다.

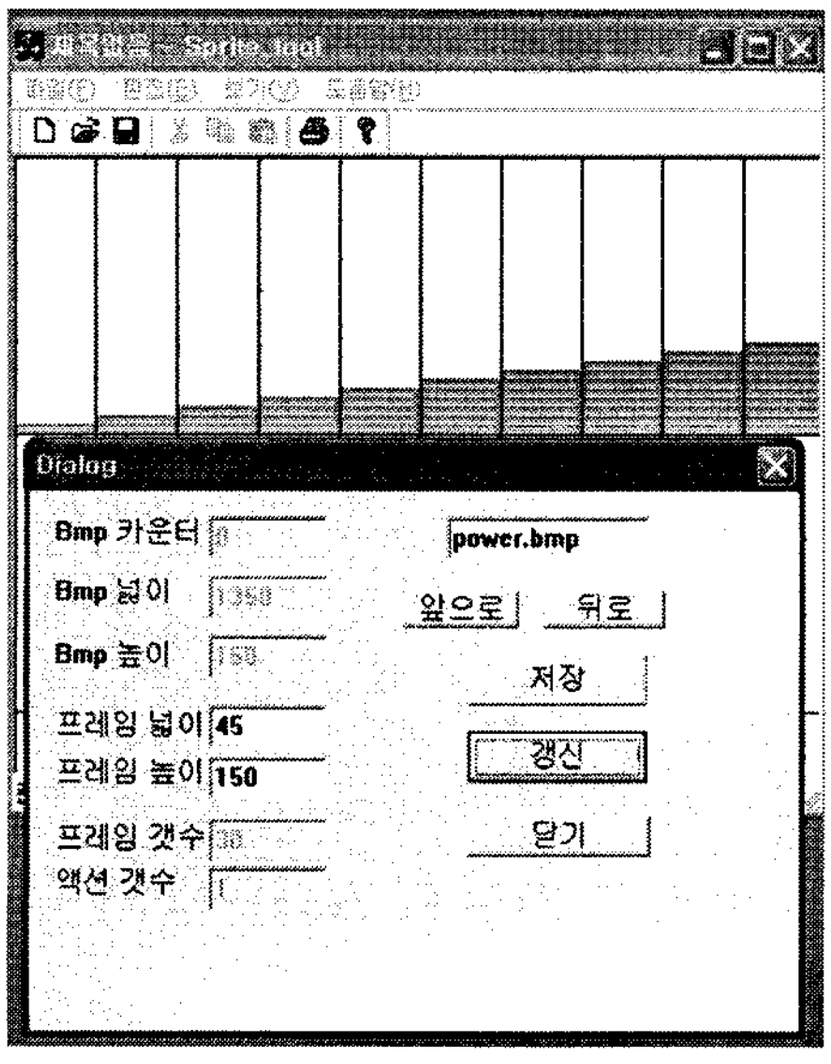

그림 5. Sprite Tool

그림 2는 스프라이트 도구를 사용하여 비트맵 이미지의 속성 값을 파일로 만드는 예를 보인다.

스프라이트 클래스는 볼링게임에서 쓰여질 클래스로서 다양한 함수들이 존재 한다. 스프라이트 도구에서 제작된 이미지 속성 파일들을 읽어드려 동적으로 속성들을 할당하고, 각 이미지 들을 업데이트시키며 출력에 필요한 함수를 가지고 있다.

1) 비트맵이미지 로딩

스프라이트 도구에서 만들어진 파일을 스프라이트 클래스에서 로딩하여 비트맵이미지 파일이름, 프레임 개수, 프레임의 폭, 높이 등을 동적으로 할당하여 메모리에 적재한다. 그림 3은 비트맵이미지의 스프라이트 속성파일을 읽어오는 구문이다.

```
int CSprite::Init()
{
 if((spr_file=fopen("resource/bowling.spr")
 fread(&bitmapindex,sizeof(int),1,spr_file);
 sprite =  new SPRITE[bitmapindex];
 m_pTexture = new TEXTURE[bitmapindex];
 m_Imaginfo = new IMAGE_INFO[bitmapindex];
 Drawrect = new RECT[bitmapindex];
 for(int i = 0;i<bitmapindex;i++)
 {
  fread(&sprite[i],sizeof(SPRITE),1,spr_file);
  m_pTexture[i]                    = NULL;
 }
 fclose(spr_file);
}
```

그림 3. 스프라이트 속성 로딩 구문

2) 비트맵이미지 출력

DirectX는 원래 3D 그래픽을 사용하기 위한 SDK이나, 2D그래픽을 사용할 수 있도록 지원해주는 클래스 LPD3DXSPRITE가 있어서 이 클래스를 이용하여 2D그래픽을 출력할 수 있다. 비트맵 출력은 Draw함수를 사용하여 비트맵이미지의 출력할 위치, 비트맵이미지의 소스좌표값 등으로 이미지를 출력한다. 그림 4는 비트맵이미지를 출력하는 구문이다.

```
int CSprite::Render()
{
 //angle
 Draw(angle_tx,&rt1,2D(4,90),&2D(641,210));
 //power
 Draw(m_pTexture[0],&Drawrect[0],0,2D(425,185));
 //spin_ball
 Draw(m_spinball_tx,&rt2,2D(m_spin,307));
 //pos
 Draw(m_pos_tx,&m_pos_rt,2D(60,747));
 //ball
 Draw(m_ball_tx,&m_ball_rt,&m_ball_pos);

 return 1;
}
```

그림 4. 스프라이트 이미지 출력 구문

스프라이트는 크게 2가지로 구현되어 있는데 이미지를 변경 하는 업데이트와 이미지를 출력하는 출력부분이 있다.

1) 2D 스프라이트 업데이트

sprite의 업데이트 방법은 게임 내의 루프를 돌면서 FPS(Frame per Sec)를 체크하여 일정한 값이 되면 프레임을 바꾸는 방법을 취하고 있다.

FPS를 구하는 방법은 매회 게임의 루프를 돌 때마다 timeGetTime()함수를 이용하고 그전의 시간 값의 차로 부터 FPS를 구한다.

그림 5는 main업데이트 구문과 그림 6은 manager 업데이트 구문을 나타낸다.

```
HRESULT CMain::FrameMove()
{
 manager.GameLoop();
 return 1;
}
```

그림 5. main 업데이트 구문

```
int CBowlingManager::GameLoop()
{
 if(GameStart          == true)   //게임 시작
 {
  input.FrameMove();
  // 엔터키 누르면 실행
  if(input.KeyDown(VK_RETURN))
  {
   m_dTimeEnd       = timeGetTime();
   if( GameTurn == true) // 나의 턴
   {
    MySpriteLoop();
   }
   else
   {
    EnemySpriteLoop();
   }
  }
 return 1;
}
```

그림 6. manager 업데이트 구문

2) 스프라이트 출력

스프라이트 출력 방법은 FPS를 체크하여 프레임들의 변화가 일어나야 할 경우에 게임화면을 갱신한다. 스프라이트쪽과 게임 처리 쪽과 맞물려서 게임을 진행 할 수 있게 된다. 그림 7은 main 출력 구문, 그림 8은 manager 출력 구문을 나타낸다.

```
HRESULT CMain::Render()
{
 m_pd3dDevice->Clear(); //화면 지움
 m_pd3dDevice->BeginScene();

 m_pd3dSprite->Begin();
 manager.Render();
 m_pd3dSprite->End();
 m_pd3dDevice->EndScene();
 return 1;
}
```

그림 7. main 출력 구문

```
int CBowlingManager::Render()
{
 image.Render();       //배경화면 출력
 sprite.Render();      //스프라이트 출력
 return 1;
}
```

그림 8. manager 출력 구문

Ⅳ. 2D 스프라이트를 이용한 볼링게임 구현

볼링 게임은 사용자로부터 winmain이 메시지 루프를 돌면서 event를 받는다. 받은 event는 manager로 전송해준다. 이 event는 manager가 이벤트 처리기를 통하여 엔진부와 처리부로 각각 분리해준다. 그림 9는 게임구조도를 나타낸다.

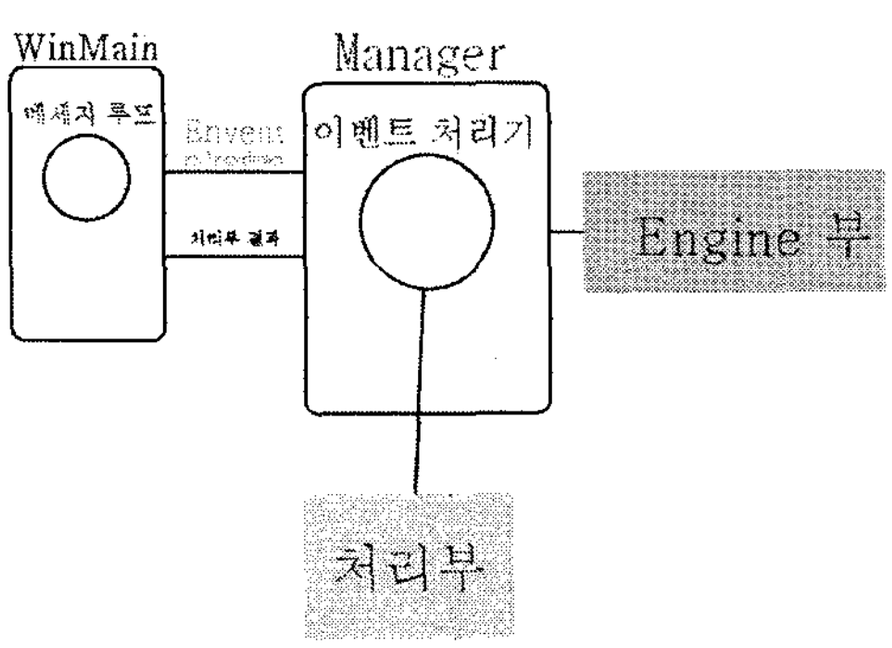

그림 9. 게임구조도

그림 10은 엔진부를 나타낸 것으로서, 엔진부는 물리엔진, 네트워크엔진, 스프라이프엔진, UI 엔진으로 나뉜다. 물리엔진은 볼링게임에 들어가는 물리부분을 관리하며 manager에 의해 핸들링된다. 네트워크엔진은 서버의 ip에 접속하여 클라이언트와 통신을 하게 된다. 스프라이트엔진은 스프라이트에 관련한 정보와 게임의 출력부를 가진다.

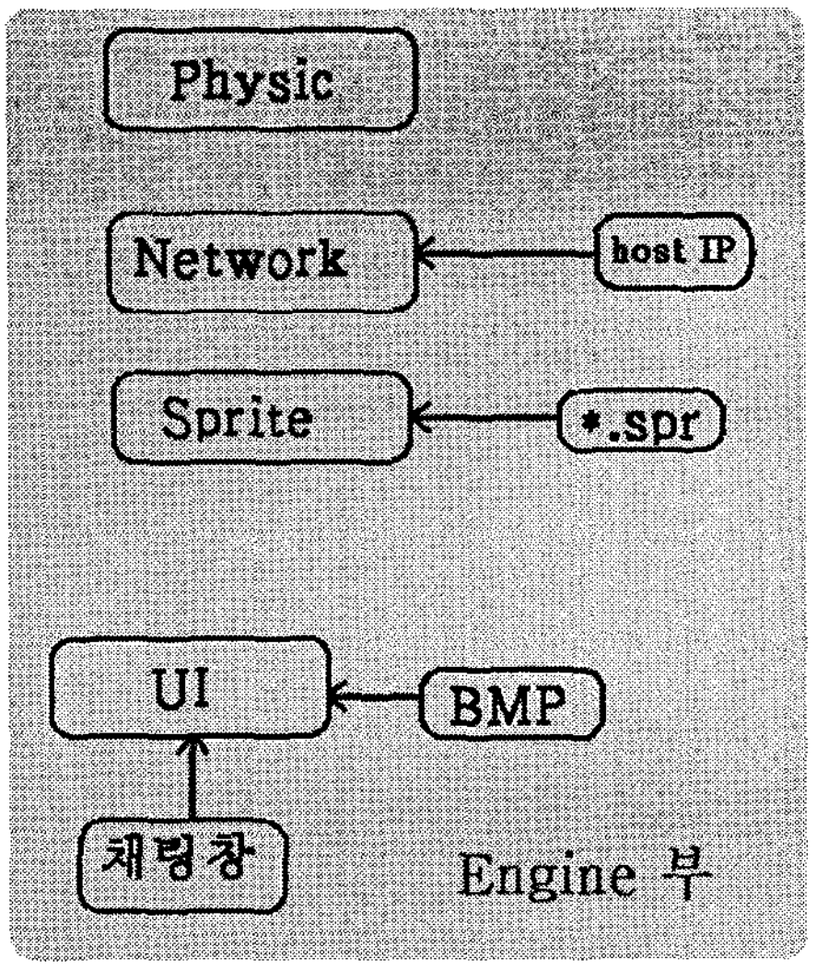

그림 10. 엔진부

그림 11은 처리부를 나타낸 것으로서, 처리부는 manager로부터 받은 event를 처리함으로서 게임을 진행하게 된다. 처리부는 게임의 로직부분으로서 manager로부터 핸들링 된다.

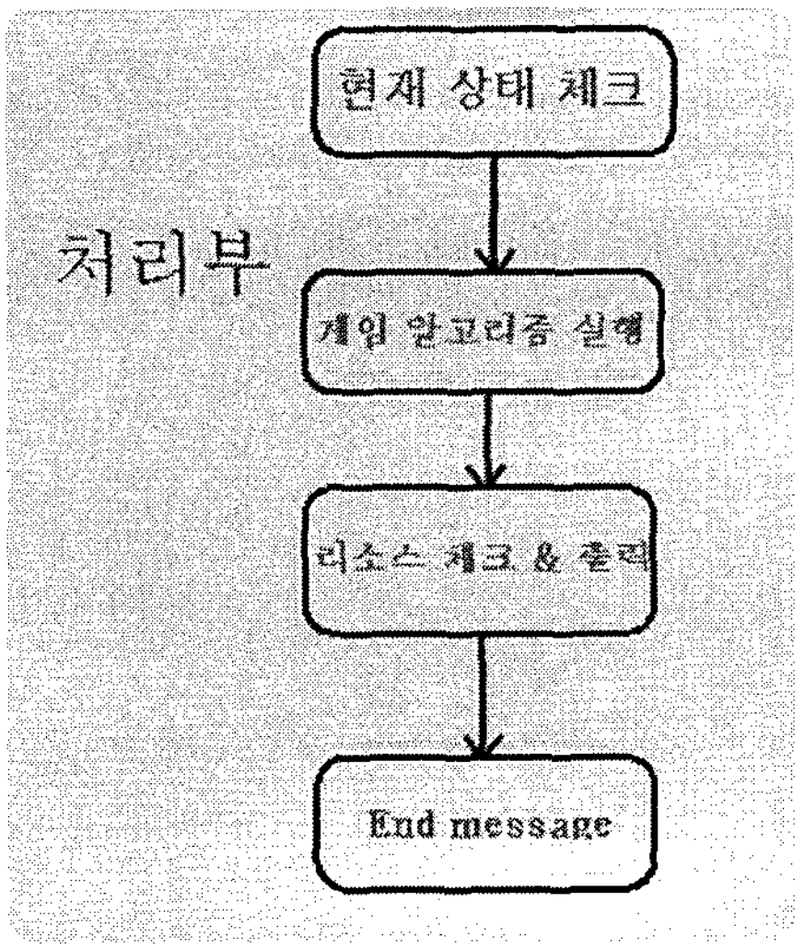

그림 11. 처리부

V. 결론

본 논문에서는 2D 스프라이트를 이용하여 게임에 적용 및 사용 하는 방법을 볼링게임에 적용하여 통해 구현하였다. 2D 스프라이트를 사용하기 위하여 스프라이트클래스를 구현하여 그에 해당하는 다양한 함수를 구현 했고, 스프라이트 도구을 제작하여 스프라이트클래스를 좀 더 효율적으로 사용할 수 있도록 구현하였다.

참고문헌

[1]최현호 역, "Directx 9를 이용한 3D Game프로그래밍 입문", 정보문화사, 2004.
[2]http://terms.naver.com/item.nhn?dirId=200&docId=9548.
[3]http://register.itfind.or.kr/Report/200301/IITA/IITA-1108/IITA-1108.pdf.