Chapter 08

슈팅 게임 디자인

(Shooting Game Design)

Contents

8.1 미사일 + 적

❖ 미사일과 적과의 충돌

◆ 폭발 애니메이션 추가

- ✓ SpriteAnimation을 상속받아 EffectExplosion 클래스 생성
- ✓ 폭발용 스프라이트 이미지

- ✓ 기본적인 코드 구성
  - ▪ 미사일과 적이 충돌했을 때 GameState 클래스의 CheckCollision 메서드에서 나타낼 효과
  - ▪ EffectExplosion 클래스의 생성자에서 이 효과를 생성할 위치를 인자로 받음

8. 1 미사일 + 적

```
public class EffectExplosion extends SpriteAnimation  {
    public EffectExplosion (int x, int y)  {
            super (AppManager.getInstance( ).getBitmap (R.drawable. explosion);
            this .InitSpriteData (104, 66, 3, 6);

            m_x = x;
            m_y = y;
    }
}
```

8.1 미사일 + 적

❖ 미사일과 적과의 충돌 (cont.)

◆ 폭발 애니메이션 추가

✓ 이 폭발 효과를 가질 리스트를 GameState 클래스에 멤버 변수로 추가하고, 화면에 그리고 업데이트하는 코드를 작성

8. 1 미사일 + 적

```
public class GameState extends IState  {
    … …
    ArrayList<EffectExplosion> m_explist = new ArrayList<EffectExplosion>( );
    … …
    @Override
    public void Render (Canvas canvas)  {
            m_background .Draw (canvas);
            for (Missle pms : m_pmslist)  {    pms.Draw(canvas);  }
            for (Missle enemms : m_enemmslist)  {  enemms.Draw(canvas);  }
            for (Enemy enem : m_enemlist)  {  enem.Draw(canvas);  }
            for (EffecExplosion exp : m_explist)  {  exp.Draw(canvas);  }
            m_player .Draw(canvas);
            m_keypad .Draw(canvas);
            … …
    }
```

8. 1 미사일 + 적

```
    @Override
    public void Update ( )  {
            long gameTime = System.currentTimeMillis( );
            m_player .Update (gameTime);
            m_background .Update (gameTime);
            … …
            for (int i = m_explist .size( ) – 1; i >= 0; i--)  {
                EffectExplosion exp = m_explist .get(i);
                exp.Update(gameTime);
            }
            MakeEnemy( );
            CheckCollision( );
    }
}
```

8.1 미사일 + 적

❖ 미사일과 적과의 충돌 (cont.)

◆ 폭발 애니메이션 추가

- ✓ 폭발 효과를 생성할 위치에서 생성
- ✓ 적과 미사일 혹은 적의 미사일과 플레이어가 충돌하는 경우 폭발 효과를 추가
- ✓ 객체가 사라지는 코드에 폭발 효과를 생성하는 코드 추가

8. 1 미사일 + 적

```
public class GameState extends IState  {
    … …
    public void CheckCollision( )  {
            for (int i = m_pmslist .size( ) – 1; i >= 0; i--)  {
                for (int j = m_enemlist .size( ) – 1; j >= 0; j--)  {
                        if (CollisionManager.CheckBoxToBox( m_pmslist .get(i). m_BoundBox,
                                                            m_enemlist .get(j). m_BoundBox))  {
                            m_explist .add( new EffectExplosion ( m_enemlist .get(j).GetX( ),
                                                                  m_enemlist .get(j).GetY( )));
                            m_pmslist .remove(i);
                            m_enemlist .remove(j);

                            return;
                        }
                }
            }
```

8. 1 미사일 + 적

```
            for (int i = m_enemslist .size( ) – 1; i >= 0; i--)  {
                if (CollisionManager.CheckBoxToBox( m_player. m_BoundBox,
                                                    m_enemlist .get(i). m_BoundBox))  {
                        m_explist .add( new EffectExplosion ( m_enemlist .get(j).GetX( ),
                                                              m_enemlist .get(j).GetY( )));
                        m_enemlist .remove(i);
                        m_player .destroyPlayer( );
                        i f (m_player .getLife( ) <= 0)
                            AppManager.getInstance( ).getActivity( ).finish( );
                }
            }
            … …
    }
}
```

8.1 미사일 + 적

❖ 미사일과 적과의 충돌 (cont.)

◆ 폭발 애니메이션 추가

- ✓ 컴파일하고 실행
- ✓ 문제점
  - 폭발 이미지가 계속 해서 남아 있음
  - SpriteAnimation 클래스의 확장성 필요
- ✓ SpriteAnimation 클래스 확장
  - 애니메이션의 반복 여부를 판별할 변수와 애니메이션이 종료되었다는 정보를 저장할 변수 추가

8. 1 미사일 + 적

```
public class SpriteAnimation extends GraphicObject  {
    … …
    protected boolean mbReplay = true;
    protected boolean mbEnd = false;
    … …
    public void Update ( long gameTime )  {
            if (!mbEnd)  {
                if ( gameTime > mFrameTimer + mFPS )  {
                        mFrameTime = gameTime;
                        mCurrentFrame += 1;
                        if (mCurrentFrame >= mNoOfFrames)  {
                            if (mbReplay)  mCurrentFrame = 0;
                            else                mbEnd = true;
                        }
                }
            }
            mSRectangle.left = mCurrentFrame * mSpriteWidth;
            mSRectangle.right = mSRectangle.left + mSpriteWidth;
    }
}
```

8.1 미사일 + 적

❖ 미사일과 적과의 충돌 (cont.)

◆ 폭발 애니메이션 추가

- ✓ SpriteAnimation 클래스 확장
  - Update 메서드로 이동해서 mbReplay가 true일 때만 마지막 프레임까지 도달했을 때 다시 첫 번째 프레임으로 바꿔 애니메이션을 반복하게 함
  - true가 아닐 때 애니메이션이 종료되면 mbEnd를 true로 변경해서 애니메이션이 종료되었다는 정보를 저장
- ✓ EffectExplosion 클래스에서 mbReplay를 false로 변경해서 애니메이션 반복하지 않게 함

8. 1 미사일 + 적

```
public class EffectExplosion extends SpriteAnimation  {
    public EffectExplosion (int x, int y)  {
            super (AppManager.getInstance( ).getBitmap (R.drawable. explosion);
            this .InitSpriteData (104, 66, 3, 6);

            m_x = x;
            m_y = y;

            mbReplay = false;
    }
}
```

8.1 미사일 + 적

❖ 미사일과 적과의 충돌 (cont.)

◆ 폭발 애니메이션 추가

- ✓ 컴파일하고 실행
- ✓ 폭발 효과가 생성된 뒤에 반복되지 않고 사라짐
- ✓ 디버깅으로 메모리 확인하면 폭발 애니메이션은 남아 있음
  - ▪ GameState 클래스의 Update 메서드에 폭발 효과들의 리스트를 업데이트하면서 폭발 효과의 애니메이션 종료시 리스트에서 제거하는 코드 추가

8. 1 미사일 + 적

```
public class SpriteAnimation extends GraphicObject  {
    … …
    public boolean getAnimationEnd( )  {
            return mbEnd;
    }
}
```

8. 1 미사일 + 적

```
public class GameState extends IState  {
    … …
    @Override
    public void Update ( )  {
            … …
            for (int i = m_explist .size( ) – 1; i >= 0; i--)  {
                EffectExplosion exp = m_explist .get(i);
                exp.Update(gameTime);
                if (exp.getAnimationEnd( ))  m_explist .remove(i);
            }
            MakeEnemy( );
            CheckCollision( );
    }
```

8.2 아이템

- 대표적 아이템
  - 점수 아이템
  - 파워 아이템
  - 생명 아이템
- 아이템 구현
  - 아이템을 위한 슈퍼 클래스
    - ✓ Item 클래스 생성
    - ✓ 깜박이는 애니메이션을 위해 SpriteAnimation 상속
    - ✓ 아이템과 플레이어간의 충돌 처리 필요
      - 생성자를 만들고, 충돌 처리와 마찬가지로 충돌 상자 값을 추가하고, Update 메서드를 재정의해서 충돌 상자를 지속적으로 업데이트

8.2 아이템

```
public class Item extends SpriteAnimation  {
    Rect m_BoundBox = new Rect( );
    public Item (Bitmap bitmap)  {
            super (bitmap);
    }
    @Override
    public void Update ( long gameTime )  {
            super .Update(gameTime);
            m_BoundBox .set ( m_x, m_y, m_x + 51, m_y + 51 );
    }
}
```

8.2 아이템

❖ 아이템 구현 (cont.)

◆ 메서드 구현

✓ 플레이어가 아이템을 먹을 때, 즉 충돌했을 때 실행할 메서드 추가

```
public class Item extends SpriteAnimation {
    … …
    void GetItem( ) }
    }
}
```

8.2 아이템

❖ 아이템 구현 (cont.)

◆ 점수 아이템 구현

- ✓ 점수 기능 추가
- ✓ GameState 클래스에 멤버 변수로 추가하는 정도로 처리

```
public class GameState extends IState  {
    … …
    private int m_score = 0;
    … …
}
```

8.2 아이템

❖ 아이템 구현 (cont.)

◆ 점수 아이템 구현

✓ ItemAddScore 클래스 생성

- 적이 파괴되고 나서 생성되는 아이템이니 적이 파괴된 위치 값을 생성자의 인자로 받아오게 함

```
public class ItemAddScore extends Item  {
    public ItemAddScore (int x, int y)  {
            super (AppManager.getInstance( ).getBitmap(R.drawable. item1));
            this .InitSpriteData(51, 51, 3, 4);

            m_x = x;
            m_y = y;
    }
}
```

8.2 아이템

❖ 아이템 구현 (cont.)

◆ 점수 아이템 구현

✓ GetItem 메서드 재정의

```
public class ItemAddScore extends Item  {
    … …
    @Override
    void GetItem( )  {
            GameState.getInstance( ). m_score += 100;
    }
}
```

8.2 아이템

❖ 아이템 구현 (cont.)

◆ 점수 아이템 구현

✓ GameState 클래스에 ArrayList 추가

```
public class GameState extends IState {
    … …
    ArrayList<Item> m_itemlist = new ArrayList<Item>( );
    … …
}
```

✓ GameState 클래스의 Render와 Update 메서드에서 아이템을 그려주고, 업데이트하는 코드 추가

8.2 아이템

```
public class GameState extends IState  {
    … …
    @Override
     public void Render (Canvas canvas)  {
              m_background .Draw (canvas);
              for (Missle pms : m_pmslist)  {    pms.Draw(canvas);  }
              for (Missle enemms : m_enemmslist)  {  enemms.Draw(canvas);  }
              for (Enemy enem : m_enemlist)  {  enem.Draw(canvas);  }
              for (EffecExplosion exp : m_explist)  {  exp.Draw(canvas);  }
              for (Item item : m_itemlist)  {  item.Draw(canvas);  }
              m_player .Draw(canvas);
              m_keypad .Draw(canvas);
              … …
    }
```

8.2 아이템

```
public class GameState extends IState  {
    … …
    @Override
    public void Update ( )  {
            … …
            for (int i = m_itemlist .size( ) – 1; i >= 0; i--)  {
                Item item = m_itemlist .get(i);
                item.Update(gameTime);
            }
            MakeEnemy( );
            CheckCollision( );
    }
}
```

8.2 아이템

❖ 아이템 구현 (cont.)

◆ 점수 아이템 구현

- ✓ CheckCollision 메서드에서 점수 아이템을 생성하는 코드 작성
- ✓ 컴파일하고 실행

8.2 아이템

```
public class GameState extends IState  {
    … …
    public void CheckCollision( )  {
            for (int i = m_pmslist .size( ) – 1; i >= 0; i--)  {
                for (int j = m_enemlist .size( ) – 1; j >= 0; j--)  {
                        if (CollisionManager.CheckBoxToBox( m_pmslist .get(i). m_BoundBox,
                                                            m_enemlist .get(j). m_BoundBox))  {
                            m_explist .add( new EffectExplosion ( m_enemlist .get(j).GetX( ),
                                                                  m_enemlist .get(j).GetY( )));
                            m_itemlist .add( new ItemAddScore ( m_enemlist .get(j).GetX( ),
                                                                m_enemlist .get(j).GetY( )));
                            m_pmslist .remove(i);
                            m_enemlist .remove(j);

                            return;
                        }
                }
            }
```

8.2 아이템

❖ 아이템 구현 (cont.)

◆ 점수 아이템 구현

✓ 점수 아이템에 대한 스크롤

```
public class Item extends SpriteAnimation  {
    … …
    public boolean bOut = false;
    … …
    @Override
    public void Update ( long gameTime )  {
            super .Update(gameTime);
            m_y += 2;
            if (m_y > 350)  bOut = true;
            m_BoundBox .set ( m_x, m_y, m_x + 51, m_y + 51 );
    }
}
```

8.2 아이템

❖ 아이템 구현 (cont.)

◆ 점수 아이템 구현

✓ 화면을 벗어난 아이템 제거 ; 컴파일하고 실행

```
public class GameState extends IState  {
    … …
    @Override
    public void Update ( )  {
            … …
            for (int i = m_itemlist .size( ) – 1; i >= 0; i--)  {
                Item item = m_itemlist .get(i);
                item.Update(gameTime);
                if (item. bOut == true)  m_itemlist .remove(i);
            }
            MakeEnemy( );
            CheckCollision( );
    }
}
```

8.2 아이템

❖ 아이템 구현 (cont.)

◆ 점수 아이템 구현

- ✓ 아이템과 플레이어와의 충돌 처리

8.2 아이템

```
public class GameState extends IState  {
    … …
    public void CheckCollision( )  {
            … …
            for (int i = m_itemlist .size( ) – 1; i >= 0; i--)  {
                if (CollisionManager.CheckBoxToBox( m_player.m_BoundBox,
                                                    m_itemlist .get(i). m_BoundBox))  {
                        m_itemlist .get(i).GetItem( );
                        m_itemlist .remove(i);
                }
            }
    }
}
```

8.2 아이템

❖ 아이템 구현 (cont.)

◆ 점수 아이템 구현

✓ 점수 화면 출력

```
public class GameState extends IState  {
    … …
    @Override
     public void Render (Canvas canvas)  {
            … …
            canvas.drawText (“점수 : “ + String.valueOf( m_score ), 0, 40, p);
    }
```

✓ 컴파일하고 실행

8.3 디자인적인 요소들

❖ 그 외 디자인적인 요소들

◆ 다른 게임과의 비교를 통한 추가

- ✓ 지금까지 게임을 좀 더 게임답게 만들어주는 요소들을 추가해보았다.
- ✓ 다른 슈팅 게임을 플레이 해보며 우리 게임에 없는 요소들을 파악하고, 각자의 역량에 따라 추가해보자.