Chapter 06

슈팅 게임 개발 (Basic)

(Shooting Game - Basic 1of2)

Contents

6.1 게임 상태 추가하기

❖ 슈팅 게임 기본 요소

- ◆ 플레이어
- ◆ 적
- ◆ 미사일

❖ 패키지 생성

- ◆ New → Package
  - ✓ org.Game 패키지 생성
  - ✓ Framework과 같은 폴더내 생성
    - ▪ ex) org.Framework / org.Game

6.1 게임 상태 추가하기

❖ 게임 상태 추가하기

◆ 게임 상태를 위한 GameState 클래스

✓ Framework의 IState 인터페이스를 상속

```
public class GameState implements IState {
    // 멤버 변수 추가
    @Override
    public void Destroy( ) { }
    @Override
    public void Init( ) { }
    @Override
    public void Render(Canvas canvas ) { }
 @Override
    public void Update( ) { }
 @Override
    public void onKeyDown( int keyCode, KeyEvent event ) { return false; }
 @Override
    public void Destroy(MotionEvent event) { return false; }
}
```

6.2 플레이어 클래스 제작하기

❖ 플레이어 클래스 추가하기

◆ Player 클래스 추가

✓ SpriteAnimation 클래스 상속

```
package org.Game;

import org.Framework.SpriteAnimation;
import android.graphics.Bitmap;

public class Player extends SpriteAnimation  {
    public Player ( Bitmap bitmap)  {
            super (bitmap);
    }
}
```

6.2 플레이어 클래스 제작하기

❖ 플레이어 클래스 추가하기

(cont.)

◆ Player 클래스 추가

✓ 애니메이션 추가

```
public class Player extends SpriteAnimation  {
    public Player ( Bitmap bitmap)  {
            super (bitmap);
            // 애니메이션 정보 설정
            this .InitSpriteData(104, 62, 3, 6);
    }
}
```

6.2 플레이어 클래스 제작하기

❖ 플레이어 클래스 추가하기 (cont.)

◆ Player 클래스 추가

✓ 프레임워크의 게임 뷰에 Player 클래스 추가

```
public class GameState implements IState  {
    // 멤버 변수 추가할 곳
    private Player m_player;
    … …
}
```

6.2 플레이어 클래스 제작하기

❖ 플레이어 클래스 추가하기 (cont.)

◆ Player 클래스 추가

✓ 생성자에서 인스턴스화하고, 그림처리와 애니메이션 업데이트 메서드 호출

✓ 컴파일하고 실행

6. 2 플레이어 클래스 제작하기

```
public class GameState implements IState  {
    … …
    @Override
    public void Init( )  {
            m_player = new Player(AppManager.getInstance( ).getBitmap
                        (R.drawable. player));
    }
    @Override
    public void Render(Canvas canvas)  {
            m_player .Draw (canvas);
    }
    @Override
    public void Update( )  {
            long GameTime = System.currentTimeMillis( );
            m_player .Update(GameTime);
    }
    … …
}
```

6.2 플레이어 클래스 제작하기

❖ 플레이어 클래스 추가하기 (cont.)

◆ 플레이어 위치 변경

✓ Player.java 에서 생성자에서 SetPosition 메서드 호출

```
public class Player extends SpriteAnimation {
    public Player ( Bitmap bitmap) {
            super (bitmap);
            // 애니메이션 정보 설정
            this .InitSpriteData(104, 62, 3, 6);
            // 초기 위치 값을 설정
            this .SetPosition (140, 380);
    }
}
```

6.2 플레이어 클래스 제작하기

❖ 플레이어 클래스 추가하기 (cont.)

◆ 키 입력 처리

✓ GameView.java에서 키입력 처리를 위한 포커스 설정

```
public class GameView extends SurfaceView implements SufaceHolder.Callback  {
    … …
    public GameView (Context context)  {
            super (context);
            // 키 입력 처리를 받기 위해서
            setFocusable( true );
            AppManager.getInstance( ).setGameView( this );
            AppManager.getInstance( ).setResources( getResources( ));
            ChangeGameState( new GameState( ));
            getHolder( ).addCallback( this );
            _thread = new GameViewThread(getHolder( ), this );
    }
    … …
}
```

6.2 플레이어 클래스 제작하기

❖ 플레이어 클래스 추가하기 (cont.)

◆ 키 입력 처리

✓ 캐릭터 이동 코드 작성

```
public class GameState implements IState  {
    … …
    @Override
    public boolean onKeyDown( int keyCode, KeyEvent event)  {
            // 키 입력에 따른 플레이어 이동
            int x = m_player .GetX( );
            int y = m_player .GetY( );
```

6. 2 플레이어 클래스 제작하기

```
            if (keyCode == KeyEvent. KECODE_DPAD_LEFT)  // 왼쪽
                m_player .SetPosition( x-1, y );
            if (keyCode == KeyEvent. KECODE_DPAD_RIGHT)  // 오른쪽
                m_player .SetPosition( x+1, y );
            if (keyCode == KeyEvent. KECODE_DPAD_UP)  // 위
                m_player .SetPosition( x, y-1 );
            if (keyCode == KeyEvent. KECODE_DPAD_DOWN)  // 아래
                m_player .SetPosition( x, y+1 );

            return true;
        }
        … …
    }
```

✓ 컴파일 하고 실행

6.3 배경 클래스 제작하기

❖ 배경 클래스 제작하기

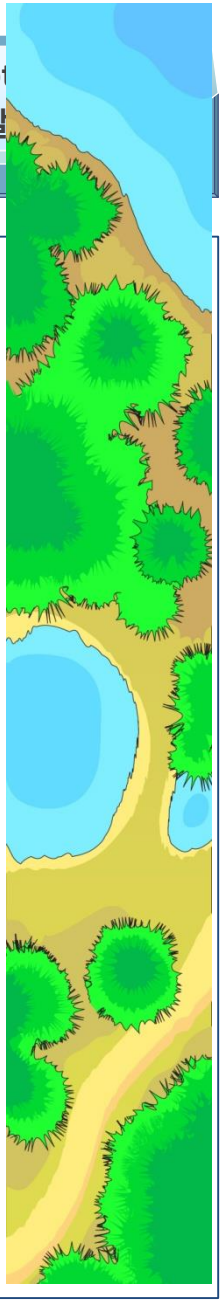
◆ BackGround 배경 클래스 추가

✓ GraphicObject를 상속

```
public class BackGround extends GraphicObject {
    public BackGround( ) {
            super (AppManager.getInstance( ).getBitmap
                    (R.drawable. background2));
    }
}
```

✓ GameState 클래스에서 BackGround 클래스를 멤버변수로 추가하고, 인스턴스화하여 배경을 그려주는 코드 작성

✓ 컴파일하고 실행



6.3 배경 클래스 제작하기

```
public class GameState implements IState  {
    private BackGround m_background;
    … …
    @Override
    public void Init( )  {
            // 키 입력에 따른 플레이어 이동
            m_player = new Player(AppManager.getInstance( ).getBitmap
                    (R.drawable .player);
            m_background = new BackGround( );
    }
    @Override
    public void Render(Canvas canvas )  {
            m_background .Draw(canvas);
            m_player .Draw(canvas);
    }
    … …
}
```

6.3 배경 클래스 제작하기

❖ 스크롤링 구현

◆ 배경 이미지를 이동

- ✓ 플레이어 캐릭터가 위로 이동하는 느낌을 표현하기 위해 배경 이미지를 지속적으로 Y 값을 증가
- ✓ 먼저, 그림의 초기 시작 지점을 설정하고 Y 값을 증가하면서 움직이는 느낌을 주는 부분을 Update 메서드에 작성

6.3 배경 클래스 제작하기

```
public class BackGround extends GraphicObject  {
    static final float SCROLL_SPEED = 0.2f;
    private float m_scroll = -2000 + 480;

    public BackGround( )  {
            super (AppManager.getInstance( ).getBitmap
                    (R.drawable. background2));
            SetPosition(0, (int) m_scroll);
    }
    void Update( long GameTime)  {
            m_scroll = m_scroll + SCROLL_SPEED;
            SetPosition(0, (int) m_scroll);
    }
}
```

6.3 배경 클래스 제작하기

❖ 스크롤링 구현 (cont.)

◆ 배경 이미지를 이동

✓ GameState 클래스에서 BackGround의 Update 메서드를 실행하는 코드 추가

```
public class GameState implements IState  {
    … …
    @Override
    public void Update( )  {
            long GameTime = System.currentTimeMillis( );
            m_player .Update(GameTime);
            m_background .Update(GameTime);
    }
    … …
}
```

6.3 배경 클래스 제작하기

❖ 스크롤링 구현 (cont.)

◆ 스크롤링

✓ 무한 스크롤링 방식 : 시작과 끝이 매끄럽게 이어지는 이미지일 때 반복해서나타냄

✓ 스탑 스크롤 방식 : 스크롤의 끝에 가면 더 이상 스크롤링을 하지 않음

→ 여기서는 스탑 스크롤 방식 사용

```
public class BackGround extends GraphicObject  {
    … …
    void Update( long GameTime)  {
            m_scroll = m_scroll + SCROLL_SPEED;
            if ( m_scroll >=0)  m_scroll = 0;
            SetPosition(0, (int) m_scroll);
    }
}
```

6.3 배경 클래스 제작하기

❖ 스크롤링 구현 (cont.)

◆ 시차 스크롤링 (Parallax Scrolling)

- ✓ 물체와 카메라의 거리에 따라 물체의 이동하는 속도가 달라 보이는 효과
- ✓ 배경은 레이어를 2개 사용
- ✓ 첫 번째 레이어는 배경화면 그대로, 두 번째 레이어는 다른 이미지 사용

```
public class BackGround extends GraphicObject {
    static final float SCROLL_SPEED = 0.2f;
    private float m_scroll = -2000 + 480;

    Bitmap m_layer2;
    static final float SCROLL_SPEED_2 = 0.2f;
    private float m_scroll_2 = -2000 + 480;
    ... ...
}
```

6.3 배경 클래스 제작하기

❖ 스크롤링 구현 (cont.)

◆ 시차 스크롤링 (Parallax Scrolling)

```
public class BackGround extends GraphicObject {
   … …
    public BackGround( ) {
            super (AppManager.getInstance( ).getBitmap
                    (R.drawable. background2));
            m_layer2 = AppManager.getInstance( ).getBitmap
                    (R.drawable. background_2);
            SetPosition(0, (int) m_scroll);
    }
    … …
}
```

의해서 시차 스크롤링에 사용할 그림을 표시

6.3 배경 클래스 제작하기

❖ 스크롤링 구현 (cont.)

◆ 시차 스크롤링 (Parallax Scrolling)

```
public class BackGround extends GraphicObject  {
    … …
    void Update( long GameTime)  {
            m_scroll = m_scroll + SCROLL_SPEED;
            if ( m_scroll >=0)  m_scroll = 0;
            SetPosition(0, (int) m_scroll);
            m_scroll_2 = m_scroll_2 + SCROLL_SPEED_2;
            if ( m_scroll_2 >=0)  m_scroll_2 = 0;
    }
    @Override
    public void Draw(Canvas canvas)  {
            canvas.drawBitmap( m_bitmap, m_x, m_y, null );
            canvas.drawBitmap( m_layer2, m_x, m_scroll_2, null );
    }
}
```

6.3 배경 클래스 제작하기

❖ 스크롤링 구현 (cont.)

◆ 시차 스크롤링 (Parallax Scrolling)

- ✓ 추가적인 확장성을 고려해서 구름 레이어가 아닌 제일 밑에 깔리는 레이어의 배경을 생성자에서 바꿀 수 있도록 수정

6.3 배경 클래스 제작하기

```
public class BackGround extends GraphicObject  {
   … …
    public BackGround( int backtype )  {
            super ( null );
            if (backtype == 0)
                m_bitmap = AppManager.getInstance( ).getBitmap
                        (R.drawable. background1)
            else if (backtype == 1)
                m_bitmap = AppManager.getInstance( ).getBitmap
                        (R.drawable. background2)
            m_layer2 = AppManager.getInstance( ).getBitmap
                        (R.drawable. background_2);
            SetPosition(0, (int) m_scroll);
    }
    … …
}
```

6.3 배경 클래스 제작하기

```
public class GameState implements IState  {
    … …
    @Override
    public void Init( )  {
            m_player = new Player(AppManager.getInstance( ).getBitmap
                    (R.drawable. player));
            m_keypad = new GraphicObject(AppManager.getInstance( ).getBitmap
                    (R.drawable. keypad));
            m_background = new BackGround( 0 );
            // 키패드 위치
            m_keypad .SetPosition(0, 460-120);
    }
    … …
}
```

6.3 배경 클래스 제작하기

❖ BackGround 클래스

◆ 전체 코드

```
public class BackGround extends GraphicObject  {
    static final float SCROLL_SPEED = 0.2f;
    private float m_scroll = -2000 + 480;

    Bitmap m_layer2;

    static final float SCROLL_SPEED_2 = 0.2f;
    private float m_scroll_2 = -2000 + 480;
```

6.3 배경 클래스 제작하기

```
public BackGround( int backtype )  {
        super ( null );
        if (backtype == 0)
            m_bitmap = AppManager.getInstance( ).getBitmap
                    (R.drawable. background1)
        else if (backtype == 1)
            m_bitmap = AppManager.getInstance( ).getBitmap
                    (R.drawable. background2)
        m_layer2 = AppManager.getInstance( ).getBitmap
                    (R.drawable. background_2);
        SetPosition(0, (int) m_scroll);
}
```

6.3 배경 클래스 제작하기

```
    void Update( long GameTime)  {
            m_scroll = m_scroll + SCROLL_SPEED;
            if ( m_scroll >=0)  m_scroll = 0;
            SetPosition(0, (int) m_scroll);
            m_scroll_2 = m_scroll_2 + SCROLL_SPEED_2;
            if ( m_scroll_2 >=0)  m_scroll_2 = 0;
    }
    @Override
    public void Draw(Canvas canvas)  {
            canvas.drawBitmap( m_bitmap, m_x, m_y, null );
            canvas.drawBitmap( m_layer2, m_x, m_scroll_2, null );
    }
}
```

6.4 적 클래스 제작하기

❖ Enemy 클래스 작성

◆ 여러 종류의 적들의 슈퍼 클래스

✓ 그래픽 객체이므로 SpriteAnimation 클래스를 상속

```
public class Enemy extends SpriteAnimation  {
    private int hp;
    private float speed;

    public Enemy(Bitmap bitmap)  {
            super (bitmap);
    }
}
```

- 한 번에 죽지 않을 수 있도록 HP 변수
- 종류별로 속도가 다를 수 있도록 Speed 변수

6.4 적 클래스 제작하기

❖ Enemy 클래스 작성 (cont.)

◆ 게임 로직에서의 적 행동

- ✓ 공격 : 미사일을 발사하면서 플레이어를 공격
- ✓ 이동 : 각자의 이동 패턴으로 전진
- ✓ 이런 행동을 Enemy 클래스의 메서드로 정의

```
public class Enemy extends SpriteAnimation  {
    … …
    void Move( )  {
            // 움직이는 로직
    }
    void Attack( )  {
            // 공격하는 로직
    }
}
```

6.4 적 클래스 제작하기

❖ Enemy 클래스 작성 (cont.)

◆ 적기 설정

» 체력 : 약함
» 속도 : 보통

» 체력 : 보통
» 속도 : 느림

» 체력 : 강함
» 속도 : 매우 느림

6.4 적 클래스 제작하기

❖ Enemy 클래스 작성 (cont.)

◆ 적 클래스 생성

✓ Enemy_1 클래스 추가 : Enemy 클래스 상속

- 생성자에서 비트맵을 불러와 Enemy 생성자의 인자로 넘겨주고, HP와 Speed의 수치를 지정
- InitSpriteData 메서드를 호출해서 적 캐릭터의 기본적인 스프라이트 애니메이션 정보를 설정

6.4 적 클래스 제작하기

❖ Enemy 클래스 작성 (cont.)

◆ 적 클래스 생성

✓ Enemy_1 클래스

```
public class Enemy_1 extends Enemy  {
    public Enemy_1( )  {
            super (AppManager.getInstance( ).getBitmap(R.drawable. enemy1));
            this .InitSpriteData(104, 62, 3, 6);
            hp = 10;
            speed = 2.5f;
    }
}
```

6.4 적 클래스 제작하기

❖ Enemy 클래스 작성 (cont.)

◆ 적 클래스 생성

✓ GameState 클래스에서 적이 화면에 보이도록 코드 추가

```
public class GameState implements IState  {
    … …
    private Enemy_1 enem = new Enemy_1( );
    … …                                    }
```

6.4 적 클래스 제작하기

❖ 적의 이동과 공격

◆ 적들이 사용하는 메서드인 Move( )와 Attack( ) 메서드 작성

✓ 기본적으로 SpriteAnimation의 Update 메서드를 상속받아 작성

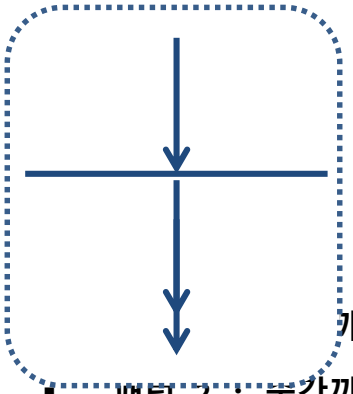
```
public class Enemy_1 extends Enemy  {
    …. …
    @Override
    public void Update( long GameTime)  {
            super .Update(GameTime);
    }
}
```

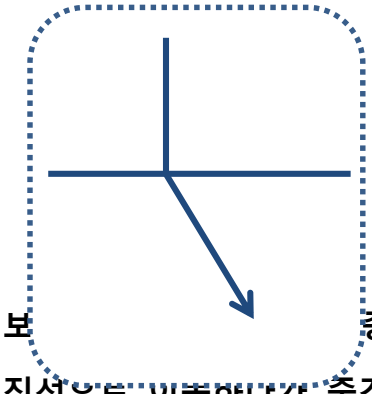
6.4 적 클래스 제작하기

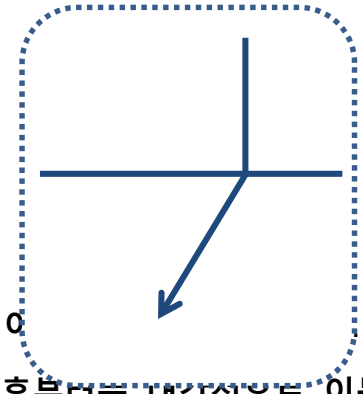
❖ 적의 이동 (cont.)

◆ 적의 이동 패턴

✓ 이동 방식의 정의







까지 보 중간 이 이동

- 패턴 2 : 중간까지 직선으로 이동하다가 중간 이후부터는 대각선으로 이동
- 패턴 3 : 두 번째와 마찬가지이지만 방향이 다른 패턴

6.4 적 클래스 제작하기

❖ 적의 이동 (cont.)

◆ 적의 이동 패턴

✓ 멤버 변수 추가

```
public class Enemy extends SpriteAnimation  {
    public static final int MOVE_PATTERN_1 = 0;
    public static final int MOVE_PATTERN_2 = 1;
    public static final int MOVE_PATTERN_3 = 2;

    protected int movetype;
    … …
}
```

6.4 적 클래스 제작하기

```
public class Enemy extends SpriteAnimation  {
    … …
    void Move( )  {
            // 움직이는 로직
            if ( movetype == MOVE_PATTERN_1 )  {
                // 첫번째 패턴
            }
            else if ( movetype == MOVE_PATTERN_2 )  {
                // 두번째 패턴
            }
            else if ( movetype == MOVE_PATTER_3 )  {
                // 세번째 패턴
            }
    }
    void Attack( )  {
            // 공격하는 로직
    }
}
```

6.4 적 클래스 제작하기

❖ 적의 이동 (cont.)

◆ 적의 이동 패턴

✓ 적의 이동 패턴 구현

- 패턴의 특징 상 모두 중간 지점을 기준으로 이동 방식이 약간씩 변형
- 이 패턴을 사용할 중간 지점을 화면 길이 /2로 지정
- 화면의 길이가 480 이므로 240이지만 실제로 안드로이드 화면의 탭 바를 포함하므로 여기서 40을 뺌
- 따라서 패턴의 움직임이 바뀌는 지점을 200으로 생각하고 코드 작성

✓ 첫 번째 패턴의 구현

6.4 적 클래스 제작하기

```
public class Enemy extends SpriteAnimation  {
    … …
    void Move( )  {
            if ( movetype == MOVE_PATTERN_1 )  {
                if ( m_y <= 200 )  m_y += speed;          // 중간지점까지 기본 속도
                else                    m_y += speed * 2;    // 중간지점이후 빠른 속도
            }
            … …
    }
    … …
}
```

6.4 적 클래스 제작하기

❖ 적의 이동 (cont.)

◆ 적의 이동 패턴

✓ Enemy의 Update) 메서드에서 Move 메서드를 호출

```
public class Enemy extends SpriteAnimation  {
    … …
    @Override
    public void Update( long GameTime )  {
            super .Update(GameTime);
            Move( );
    }
    … …
}
```

6.4 적 클래스 제작하기

❖ 적의 이동 (cont.)

◆ 적의 이동 패턴

✓ 두 번째 패턴의 구현

```
public class Enemy extends SpriteAnimation  {
    … …
    void Move( )  {
            if ( movetype == MOVE_PATTERN_1 )  {   … …  }
            else if ( movetype == MOVE_PATTERN_2 )  {
                if ( m_y <= 200 )  m_y += speed;          // 중간지점까지 일자로 이동
                else  {
                        m_x += speed;                     // 중감지점이후 대각선 이동
                        m_y += speed;
                }
            else if ( movetype == MOVE_PATTERN_3 ) {  … …  }
    }
    … …
}
```

6.4 적 클래스 제작하기

❖ 적의 이동 (cont.)

◆ 적의 이동 패턴

✓ Enemy_1 클래스의 생성자에서 movetype을 Enemy.Move_PATTERN_2로 변경하여 정상 동작 확인

```
public class Enemy_1 extends Enemy  {
    public Enemy_1( )  {
            super (AppManager.getInstance( ).getBitmap(R.drawable. enemy1));
            this .InitSpriteData(104, 62, 3, 6);
            hp = 10;
            speed = 2.5f;

            movetype = Enemy.MOVE_PATTERN_2;
    }
    …. …
}
```

6.4 적 클래스 제작하기

❖ 적의 이동 (cont.)

◆ 적의 이동 패턴

✓ 세 번째 패턴의 구현

```
public class Enemy extends SpriteAnimation  {
    … …
    void Move( )  {
            if ( movetype == MOVE_PATTERN_1 )  {   … …  }
            else if ( movetype == MOVE_PATTERN_2)  {  … …  }
            else if ( movetype == MOVE_PATTERN_3 )  {
                if ( m_y <= 200 )  m_y += speed;          // 중간지점까지 일자로 이동
                else  {
                        m_x -= speed;                     // 중감지점이후 대각선 이동
                        m_y += speed;
            }
    }
    … …
}
```

6.4 적 클래스 제작하기

❖ 적의 이동 (cont.)

◆ 적의 이동 패턴

✓ Enemy_1 클래스의 생성자에서 movetype을 Enemy.Move_PATTERN_3로 변경하여 정상 동작 확인

◆ GameState 코드

✓ 적 한마리 생성시

```
public class GameState implements IState  {
    … …
    private Enemy_1 enem = new Enemy_1( );
    … …                                    }
```

6.4 적 클래스 제작하기

❖ 적의 이동 (cont.)

◆ 적기 생성

✓ GameState 클래스에서 적의 리스트가 될 멤버 변수를 추가

```
public class GameState implements IState  {
    … …
    // private Enemy_1 enem = new Enemy_1( );
    ArrayList<Enemy> m_enemlist = new ArrayList<Enemy>( );
    … …
}
```

6.4 적 클래스 제작하기

❖ 적의 이동 (cont.)

◆ 적기 생성

- ✓ 시간이 지날때 마다 적이 등장하는 로직 추가
- ✓ GameState 클래스에 MakeEnemy 메서드와 스크롤 값을 가질 멤버 변수 추가

```
public class GameState implements IState  {
    … …
    long LastRegenEnemy = System.currentTimeMillis( );

    public void MakeEnemy( )  {

    }
    … …
}
```

6.4 적 클래스 제작하기

❖ 적의 이동 (cont.)

◆ 적기 생성

✓ MakeEnemy 메서드로 Update 메서드에서 호출

```
public class GameState implements IState  {
    … …
    @Override
    public void Update( )  {
            long GameTime = System.currentTimeMillis( );
            m_player .Update(GameTime);
            m_background .Update(GameTime);
            MakeEnemy( );
    }
    … …
}
```

6.4 적 클래스 제작하기

❖ 적의 이동 (cont.)

◆ 적기 생성

✓ 다시 MakeEnemy 메서드에서 스크롤 값에 속도 값을 추가하고, 30 정도의 스크롤 간격으로 적을 생성하는 코드 작성

```
public class GameState implements IState  {
    … …
    public void MakeEnemy( )  {
            if (System.currentTimeMillis( ) – LastRegenEnemy >= 1000 )  {
                LastRegenEnemy = System.currentTimeMillis( );

                Enemy enem = new Enemy_1( );
                enem .SetPosition(0, -60);
                enem .movetype = Enemy. MOVE_PATTERN_1;

                m_enemlist .add( enem );
            }
    }    … …  }
```

6.4 적 클래스 제작하기

❖ 적의 이동 (cont.)

◆ 적기 생성

✓ 리스트의 모든 적을 업데이트하고 화면에 그려주는 코드 작성

```
public class GameState implements IState  {
    … …
    @Override
    public void Render(Canvas canvas)  {
            m_background .Draw(canvas);
            for ( Enemy enem : m_enemlist)  {
                enem .Draw(canvas);
            }
            m_palyer .Draw(canvas);
            m_keypad .Draw(canvas);
    }
```

6.4 적 클래스 제작하기

```
    @Override
    public void Update( ) {
            long GameTime = System.currentTimeMillis( );
            m_player .Update(GameTime);
            m_background .Update(GameTime);
            for ( Enemy enem : m_enelist ) {
                enem .Update(GameTime);
            }
            MakeEnemy( );
     }
    … …
}
```

6.4 적 클래스 제작하기

❖ 적의 이동 (cont.)

◆ 적기 생성

✓ 적기의 생성이 같은 위치 → x 좌표를 랜덤하게 생성

```
public class GameState implements IState  {
    Random randEnem = new Random( );
    … …
    public void MakeEnemy( )  {
            if (System.currentTimeMillis( ) – LastRegenEnemy >= 1000 )  {
                LastRegenEnemy = System.currentTimeMillis( );

                Enemy enem = new Enemy_1( );
                enem .SetPosition( randEnem .nextInt(280), -60);
                enem .movetype = randEnem .nextInt(3);

                m_enemlist .add( enem );
            }
    } … …  }
```

6.4 적 클래스 제작하기

❖ 적의 이동 (cont.)

◆ 적기 생성

✓ 한 종류의 적 수정

- enemy2

- enemy3

6.4 적 클래스 제작하기

❖ 적의 이동 (cont.)

◆ 적기 생성

- ✓ 각 클래스 생성 (Enemy_2, Enemy_3)
- ✓ GameState 클래스의 MakeEnemy 메서드에서 객체를 생성하는 부분의 Enemy_1을 Enemy_2나 Enemy_3으로 실행
- ✓ 세 가지 적의 등장을 위해
  - MakeEnemy 메서드에서 랜덤하게 적을 생성할 수 있도록 수정

6. 4 적 클래스 제작하기

```
public class GameState implements IState  {
    Random randEnem = new Random( );
    … …
    public void MakeEnemy( )  {
            if (System.currentTimeMillis( ) – LastRegenEnemy >= 1000 )  {
                LastRegenEnemy = System.currentTimeMillis( );

                int enemtype = randEnem .nextInt(3);
                Enemy enem = null;
                if ( enemtype == 0)          enem = new Enemy_1( );
                else if ( enemtype == 1)     enem = new Enemy_2( );
                else if ( enemtype == 2)     enem = new Enemy_3( );

                enem .SetPosition( randEnem .nextInt(280), -60);
                enem .movetype = randEnem .nextInt(3);

                m_enemlist .add( enem );
            }
    }
    … …
}
```

6.4 적 클래스 제작하기

❖ 적의 이동 (cont.)

◆ 적기 생성

- ✓ 적의 다양성을 느낄 수 있으나, 일정한 패턴 형식을 보임
- ✓ 실제 게임에서은 패턴적 생성을 보완하는 알고리즘을 이용해 랜덤 클래스를 직접 제작
- ✓ 대부분의 슈팅 게임은 적들이 등장하는 타이밍과 그때마다 등장하는 적의 이동 정보 공격 패턴 등을 일일이 코드로 작성
- ✓ 하지마 두 코드의 기반은 거의 비슷