Chapter 06

슈팅 게임 개발 (Basic)

(Shooting Game - Basic 2of2)

Contents

6.5 미사일 클래스 제작하기

❖ 미사일의 부모 클래스 만들기

◆ Missile 클래스

- ✓ 적마다 미사일의 모양과 크기가 다르고, 플레이어가 발사할 미사일의 모양도 다르기 때문에 모든 미사일의 부모 클래스가 될 클래스를 하나 생성
- ✓ 미사일도 그래픽이므로 GraphicObject 클래스 상속

```
public class Missile extends GraphicObject {

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

6.5 미사일 클래스 제작하기

❖ 미사일 구현

◆ 플레이어가 발사할 미사일 클래스 생성

```
public class Missile_Player extends Missile  {

    public Missile_Player (int x, int y)  {
            super (AppManager.getInstance( ).getBitmap(R.drawable.missile_1));
            this. SetPosition(x, y);
    }
    public void Update( )  {
    }
}
```

✓ x, y 는 미사일이 생성될 위치, 즉 미사일이 생성되어 발사되는 위치

✓ ArrayList를 이용하여 플레이어의 미사일 구현

6.5 미사일 클래스 제작하기

❖ 미사일 구현 (cont.)

◆ GameState 클래스

```
public class GameState implements IState {
    … …
    ArrayList<Missile_Player> m_pmslist = new ArrayList<Missile_Player>( );
    … …
}
```

- ✓ 미사일도 한 개의 객체만 생성되는 것이 아니기 때문에 ArrayList 이용
- ✓ 스페이스바를 눌렀을 때 그 위치에서 미사일이 생성되도록 구현
- ✓ GameState 클래스의 onKeyDown 메서드

6. 5

```
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( );

            if (KeyCode == KeyEvent. KEYCODE_DPAD_LEFT)  // 왼쪽
                m_player .SetPosition(x-1, y);
            if (KeyCode == KeyEvent. KEYCODE_DPAD_RIGHT)  // 오른쪽
                m_player .SetPosition(x+1, y);
            if (KeyCode == KeyEvent. KEYCODE_DPAD_UP)  // 위
                m_player .SetPosition(x, y-1);
            if (KeyCode == KeyEvent. KEYCODE_DPAD_DOWN)  // 아래
                m_player .SetPosition(x, y+1);
            if (KeyCode == KeyEvent. KEYCODE_SPACE)  // 스페이스바
                m_pmslist .add (new Missile_Player(x,y));
            return true;
    }
    … …
}
```

6.5 미사일 클래스 제작하기

❖ 미사일 구현 (cont.)

◆ GameState 클래스

✓ 미사일을 눈에 보이는 Draw 작업

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

6.5 미사일 클래스 제작하기

❖ 미사일 구현 (cont.)

◆ GameState 클래스

✓ 미사일을 눈에 보이는 Draw 작업

- 컴파일하고 실행 → 미사일이 플레이어 좌측상단에 그려짐 → 발사위치 조절

```
public class GameState implements IState  {
    … …
    @Override
    public boolean onKeyDown (int keyCode, KeyEvent event)  {
            … …
            if (keyCode == KeyEvent. KEYCODE_SPACE)  // 스페이스 바를 눌렀을 때
                m_pmslist .add ( new Missile_Player(x+10, y) );
            return true;
    }
    … …
}
```

6.5 미사일 클래스 제작하기

❖ 미사일 구현 (cont.)

◆ GameState 클래스

✓ 미사일이 발사되는 효과

```
public class Missile_Player extends Missile  {


    … …
    public void Update( )  {
            // 미사일이 위로 발사되는 효과를 준다.
            m_y -= 2;
    }
}
```

6.5 미사일 클래스 제작하기

❖ 미사일 구현 (cont.)

◆ GameState 클래스

✓ 미사일이 발사되는 효과

- GameState 클래스의 Update 메서드로 이동해서 프레임마다 플레이어 미사일 리스트에 있는 모든 미사일의 Update 메서드가 호출되도록 작성

```
public class GameState implements IState  {
    … …
    @Override
    public void Update( )  {
            long gameTime = System.currentTimeMillis( );
            m_player .Update(gameTime);
            m_background .Update(gameTime);
            for ( Missile_Player pms : m_pmslist )    pms.Update( );
            for ( Enemy enem : m_enemlist )    enem.Update(gameTime);
            MakeEnemy( );
    }
}
```

6.5 미사일 클래스 제작하기

❖ 미사일 구현 (cont.)

◆ 미사일 발사

- ✓ 문제없이 미사일 발사?! → 미사일을 수십개씩 쏘면 강제종료됨
- ✓ 미사일이 발사되는 간격을 설정하지 않았기 때문에 미사일이 무제한으로 발사
- ✓ 이런 오류가 발생하지 않아도 게임이 진행되면서 게임 속도가 현격히 저하
- ✓ 계산량을 줄이기 위해 화면 밖을 벗어난 적과 미사일을 각 리스트에서 제거

6.5 미사일 클래스 제작하기

❖ 미사일 구현 (cont.)

◆ 최적화 구현

✓ 미사일, 적의 현재상태를 나타내는 멤버변수 하나씩 추가, 상태를 위한 상수도 추가

```
public class Missile extends GraphicObject  {
    public static final int STATE_NORMAL = 0;
    public static final int STATE_OUT = 1;
    public int state = STATE_NORMAL;
    … …
}
```

```
public class Enemy extends SpriteAnimation  {
    public static final int STATE_NORMAL = 0;
    public static final int STATE_OUT = 1;
    public int state = STATE_NORMAL;
    … …
}
```

6.5 미사일 클래스 제작하기

❖ 미사일 구현 (cont.)

◆ 최적화 구현

✓ 각각의 움직이는 로직 (Missile 클래스의 Update 메서드, Enemy 클래스의 Move 메서드)에서 화면 밖을 벗어나면 해당 객체의 state 변수 값을 STATE_OUT으로 변경

```
public class Missile extends GraphicObject {
    … …
    public void Update( ) {
            … …
            if ( m_y < 0 ) state = STATE_OUT;
    }
}
```

```
public class Enemy extends SpriteAnimation {
    … …
    void Move( ) {
            … …
            if ( m_y > 400 )    state = STATE_OUT;
    }
}
```

6.5 미사일 클래스 제작하기

❖ 미사일 구현 (cont.)

◆ 최적화 구현

- ✓ GameState의 Update 메서드에서 state 변수의 정보를 토대로 화면 밖으로 나간 객체를 각 리스트에서 제거

6. 5 [illegible]

```
public class GameState implements IState  {
    … …
    @Override
    public void Update( )  {
            long gameTime = System.currentTimeMillis( );
            m_player .Update(gameTime);
            m_background .Update(gameTime);
            for ( int i=m_pmslist .size( )-1; i >= 0; i--)  {
                Missile_Player pms = m_pmslist .get( i );
                pms .Update( );
                if ( pms. state == Missile. STATE_OUT )    m_pmslist .remove( i );
            }
            for ( int i=m_enemlist .size( )-1; i >= 0; i--)  {
                Enemy enem = m_enemlist .get( i );
                enem .Update(gameTime );
                if (enem. state == Enemy. STATE_OUT )    m_enemlist .remove( i );
            }
            MakeEnemy( );
    }
    … …
}
```

6.5 미사일 클래스 제작하기

❖ 미사일 구현 (cont.)

◆ 최적화 구현

✓ 범위 수정하여 확인

```
public class Missile extends GraphicObject  {
    … …
    public void Update( )  {
            … …
            if ( m_y < 50 )  state = STATE_OUT;
    }
}
```

```
public class Enemy extends SpriteAnimation  {
    … …
    void Move( )  {
            … …
            if ( m_y > 350 )    state = STATE_OUT;
    }
}
```

6.5 미사일 클래스 제작하기

❖ 미사일 구현 (cont.)

◆ 최적화 구현

- ✓ 아직 미사일을 많이 발사하면 애플리케이션 작동이 멈춤
- ✓ 미사일 발사 간격 조정, 최대 발사 미사일 수 설정 등으로 문제 해결 가능

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현

◆ Collision 클래스

✓ 충돌 처리에 관한 로직을 담당

```
public class CollisionManager {

}
```

◆ [illegible]

✓ 바운드 박스 (bound box), 충돌구, 충돌점 등

✓ 대표적인 바운드 박스 이용

```
public class CollisionManager {
    public static boolean CheckBoxToBox(Rect _rt1, Rect _rt2) {
            if (_rt1. right > _rt2. left && _rt1. left < _rt2. right &&
                _rt1. top > _rt2. bottom && _rt1. bottom < _rt2. top)
                return true;
            return false;
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 충돌 처리 방식

- ✓ CollisionManager를 사용해서 플레이어의 미사일과 적과의 충돌 처리를 구현
- ✓ 필요한 BoundBox 정보를 미사일과 적 클래스의 멤버 변수로 추가

```
public class Missile extends GraphicObject  {
    Rect m_BoundBox = new Rect( );
    … …
}
```

```
public class Enemy extends SpriteAnimation  {
    Rect m_BoundBox = new Rect( );
    … …
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 충돌 처리 진행

- ✓ CollisionManager를 사용해서 적과 미사일의 충돌 처리를 진행
- ✓ GameState 클래스에 적과 미사일이 충돌했는지를 지속적으로 검출하는 메서드를 작성

```
public class GameState implements IState  {
    … …
    public void CheckCollision( )  {
            for (Missile_Player pms : m_pmslist)  {
                for (Enemy enem : m_enemlist)  {
                        CollisionManager.CheckBoxToBox(pms. m_BoundBox,
                                                       enem. m_BoundBox);
                }
            }
    }
    … …
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 충돌 처리 진행

✓ 우선 GameState의 Update 메서드에서 호출

6. 6 초등 학교 구현하기

```
public class GameState implements IState  {
    … …
    @Override
    public void Update( )  {
            long gameTime = System.currentTimeMillis( );
            m_player .Update(gameTime);
            m_background .Update(gameTime);
            for (Missile_Player pms : m_pmslist)  {
                pms.Update( );
                if (pms. state == Missile. STATE_OUT)  m_pmslist .remove(pms);
            }
            for (int i = m_enemlist .size( )-1; i >= 0; i--)  {
                Enemy enem = m_enemlist .get( i );
                enem.Update(gameTime);
                if (enem. state == Enemy. STATE_OUT)  m_enemlist .remove( i );
            }
            MakeEnemy( );
            CheckCollision( );
    }
    … …
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 충돌 처리 진행

✓ 적이나 미사일이 이동할 때마다 충돌 박스의 값이 바뀌니 이를 각각의 Update 메서드에서 갱신

```
public class Missile_Player extends Missile  {
    … …
    public void Update( )  {
            // 미사일이 위로 발사되는 효과
            m_y -= 2;
            if (m_y < 50)  state = STATE_OUT;

            m_BoundBox.left = m_x;
            m_BoundBox.top = m_y;
            m_BoundBox.right = m_x + 43;
            m_BoundBox.bottom = m_y + 43;
    }
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 충돌 처리 진행

✓ Enemy_2, Enemy_3 클래스에 코드 추가

- 적의 크기 ; 62 x 104

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

            m_BoundBox .set( m_x, m_y, m_x + 62, m_y + 104);
    }
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 충돌 결과에 따른 처리

- ✓ 충돌 처리가 되면 미사일과 적이 모두 사라지는 코드 작성
- ✓ 컴파일하고 실행하여 확인

6.6 충돌 처리 구현하기

```
public class GameState implements 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_pmslist .remove(i);
                            m_enemlist .remove(j);

                            return; // 일단 루프에서 빠져나옴
                        }
                }
            }
    }
    … …
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 플레이어와 적과의 충돌 처리

- ✓ Player 클래스에 충돌 상자 생성
- ✓ 플레이어도 충돌 상자 값도 계속해서 갱신해야 하므로 Update 메서드를 수정
  - ▪ 현재 위치 값과 충돌 상자 값을 계산해서 지속적으로 값을 갱신

6.6 충돌 처리 구현하기

```
public class Player extends SpriteAnimation  {
    Rect m_BoundBox = new Rect( );
    … …
    // 프레임워크 Update에서 지속적으로 호출할 메서드
    @Override
    public void Update (long gameTime)  {
            super .Update(gameTime);
            // 움직임이 비활성화되어 있을 경우
            if (bMove)  {
                this .m_x += _dirX;
                this .m_y += _dirY;
            }
            m_BoundBox .set( m_x, m_y, m_x + 62, m_y + 104 );
    }
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 플레이어와 적과의 충돌 처리

✓ 적과의 충돌에 대한 로직 작성 ; 충돌하면 강제 종료

```
public class GameState implements IState  {
    … …
    public void CheckCollision( )  {
            …. …
            for ( int i = m_enemlist .size( )-1; i >= 0; i--)  {
                if (CollisionManager.CheckBoxToBox (
                                                m_player .m_BoundBox,
                                                m_enemlist .get(i). m_BoundBox))  {
                        System.exit(0);
                        return;
                }
            }
    }
    … …  }
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 플레이어와 적과의 충돌 처리

✓ 한번 충돌했다고 강제종료?? → 플레이어에게 생명을 추가

```
public class Player extends SpriteAnimation {
    … …
    int m_Life = 3;
    … …
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 플레이어와 적과의 충돌 처리

✓ 이 생명 값을 처리하는 메서드

```
public class Player extends SpriteAnimation {
    … …
    public int getLife( ) { return m_Life; }
    public void addLife( ) { m_Life++; }
    public void destroyPlayer( ) { m_Life--; }
    … …
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 플레이어와 적과의 충돌 처리

✓ GameState 클래스의 CheckCollision 메서드에서 플레이어의 생명 값을 내리는 destroyPalyer 메서드를 호출하고, getLife 메서드를 통해 현재 플레이어의 target이

```
public class GameState implements IState  {
    … …
    public void CheckCollision( )  {
            …. …
            for ( int i = m_enemlist .size( )-1; i >= 0; i++)  {
                if (CollisionManager.CheckBoxToBox( m_player.m_BoundBox,
                                                    m_enemlist .get( i ). m_BoundBox))  {
                        m_player .destroyPlayer( );
                        if ( m_player .getLife( ) <= 0 )  System.exit( 0 );
                }
            }
    }
    … …
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 플레이어와 적과의 충돌 처리

✓ 컴파일 하고 실행 → 한 마리의 적과 충돌하면 종료?

- 충돌한 한 마리의 적이 계속 남아 생명을 연달아 줄임
- 충돌한 적도 리스트에서 제거

6.6 충돌 처리 구현하기

```
public class GameState implements IState  {
    … …
    public void CheckCollision( )  {
            …. …
            for ( int i = m_enemlist .size( )-1; i >= 0; i++)  {
                if (CollisionManager.CheckBoxToBox( m_player.m_BoundBox,
                                                    m_enemlist .get( i ). m_BoundBox))  {
                        m_enemlist .remove( i );  // 충돌한 적도 제거
                        m_player .destroyPlayer( );
                        if ( m_player .getLife( ) <= 0 )  System.exit( 0 );
                }
            }
    }
    … …
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 플레이어와 적과의 충돌 처리

✓ 남은 생명 화면 표시

✓ 컴파일하고 실행

6.6 충돌 처리 구현하기

```
public class GameState implements IState  {
    … …
    @Override
    public void Render (Canvas canvas)  {
            m_background .Draw(canvas);
            for (Missile pms : m_pmslist)  {  pms.Draw(canvas);  }
            for (Enemy enem : m_enemlist)  {  enem.Draw(canvas);  }
            m_player .Draw(canvas);
            m_kepad .Draw(canvas);

            Paint p = new Paint( );
            p.setTextSize(20);
            p.setColor(Color. BLACK);
            canvas.drawText("남은 목숨 :"+String.valueOf( m_player .getLife( )), 0, 20, p);
    }
    … …
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 적 미사일 구현

- ✓ Missile 클래스를 상속받는 적들의 미사일 클래스 생성
- ✓ 미사일로 사용할 리소스 추가하고 생성자와 Update 메서드를 작성
- ✓ 기본적인 미사일로서의 로직은 플레이어의 미사일과 같음

6.6 충돌 처리 구현하기

```
public class Missile_Enemy extends Missile  {
    Missle_Enemy( int x, int y)  {
            super (AppManager.getInstance( ).getBitmap(R.drawable. missile_2));
            this .setPosition(x, y);
    }
    public void Update( )  {
            // 미사일이 아래로 발사되는 효과
            m_y += 2;
            if (m_y > 350)  state = STATE_OUT;

            m_BoundBox .set( m_x, m_y, m_x + 43, m_y + 43);
    }
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 적 미사일 구현

✓ 적 미사일 화면에 표시

- GameState 클래스에서 적의 미사일을 담을 리스트를 멤버 변수로 추가

```
public class GameState implements IState  {
    … …
    ArrayList<Missile> m_enemmslist = new ArrayList<Missile>( );
    … …
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 적 미사일 구현

✓ Render 메서드와 Update 메서드에서 적의 미사일을 그리고 업데이트

```
public class GameState implements IState {
    … …
    @Override
    public void Render (Canvas canvas) {
            m_background .Draw(canvas);
            for (Missile pms : m_pmslist) { pms.Draw(canvas); }
            for (Missile enemms : m_enemmslist) { enemms.Draw(canvas); }
            for (Enemy enem : m_enemlist) { enem.Draw(canvas); }
            m_player .Draw(canvas);
            m_kepad .Draw(canvas);
            … …
    }
    … …
}
```

6.6 충돌 처리 구현하기

```
public class GameState implements IState  {
    … …
    @Override
    public void Update ( )  {
            … …
            for ( int i = m_enemmslist .size( )-1; i >= 0; i--)  {
                Missile enemms = m_enemmslist .get(i);
                enemms.Update( );
                if ( enemms. state == Missile. STATE_OUT)  {  m_enemmslist .remove(i); }
            }
            … …
    }
    … …
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 적 미사일 구현

✓ 적 미사일 발사 코드

- Enemy 클래스에 이전에 발사했던 시간을 저장해서 현재 시간과 이전에 발사했던 시간을 비교해서 시간이 어느정도 흐르면 미사일을 다시 발사하도록 구현
- 시간 정보를 저장할 멤버 변수 추가

```
public class Enemy extends SpriteAnimation  {
    … …
    long LastShoot = System.currentTimeMillis( );
    … …
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 적 미사일 구현

✓ Updat 메서드에서 Attack 메서드를 호출하게 수정

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

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 적 미사일 구현

✓ Attack 메서드 구현

- 일정 간격을 두고 미사일 객체를 생성하고, GameState의 멤버 변수인 enemmlist에 추가하는 코드 작성 → Enemy 클래스에서 GameState 클래스의 멤버에 접근할 방법?
- 이를 위해 GameState를 AppManager에 추가해서 GameState를 전역 변수처럼 접근할 수 있게 해주고, GameState 생성자에서 AppManager에 자신의 값을 넘기는 코드 작성

```
public class AppManager  {
    … …
    // GameState
    public GameState m_gameState;
    … …
}
```

6.6 충돌 처리 구현하기

```
public class GameState implements IState  {
    … …
    public GameState( )  {
            AppManager.getInstance( ). m_gameState = this;
    }
    … …
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 적 미사일 구현

✓ 이제 GameState에 접근 가능

- 미사일 발사 로직 추가

```
public class Enemy extends SpriteAnimation  {
    … …
    public void Attack( )  {
            if (System.currentTimeMillis( ) – LastShoot >= 1000)  {
                LastShoot = System.currentTimeMillis( );
                // 미사일 발사 로직
                AppManager.getInstance( ). m_gameState. m_enemmslist .add(
                                                    new Missile_Enemy( m_x, m_y));
            }
    }
    … …
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 적 미사일 구현

✓ 컴파일하고 실행

- 적의 꼬리에 미사일이 보임
- 수정사항
  - » 미사일의 발사 위치
  - » 미사일의 이동 속도

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 적 미사일 구현

✓ 미사일 생성 위치 변경

```
public class Enemy extends SpriteAnimation  {
    … …
    public void Attack( )  {
            if (System.currentTimeMillis( ) – LastShoot >= 1000)  {
                LastShoot = System.currentTimeMillis( );
                // 미사일 발사 로직
                AppManager.getInstance( ). m_gameState. m_enemmslist .add(
                                        new Missile_Enemy( m_x + 10, m_y + 104));
            }
    }
    … …
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 적 미사일 구현

✓ 미사일 이동 속도 조정

```
public class Missile_Enemy extends Missile  {
    … …
    public void Update( )  {
            // 미사일이 아래로 발사되는 효과
            m_y += 4;  // 수정
            if (m_y > 350)  state = STATE_OUT;

            m_BoundBox .set( m_x, m_y, m_x + 43, m_y + 43);
    }
}
```

6.6 충돌 처리 구현하기

❖ 충돌 처리 구현 (cont.)

◆ 적 미사일과 플레이어의 충돌 처리

- ✓ GameSate 클래스의 CheckCollision 메서드에서 적 미사일과 플레이어의 충돌 처리를 구현

- ✓ v0.9

6.6 충돌 처리 구현하기

```
public class GameState implements IState  {
    … …
    public void CheckCollision( )  {
            …. …
            for ( int i = m_enemmslist .size( )-1; i >= 0; i++)  {
                if (CollisionManager.CheckBoxToBox( m_player.m_BoundBox,
                                                    m_enemmslist .get( i ). m_BoundBox))  {
                        m_enemmslist .remove( i );
                        m_player .destroyPlayer( );
                        if ( m_player .getLife( ) <= 0 )  System.exit( 0 );
                }
            }
    }
    … …
}
```