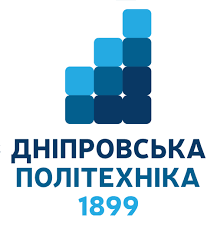
Міністерство освіти і науки України.

НТУ «Дніпровська політехніка».

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ.

Кафедра програмного забезпечення комп'ютерних систем. 

# З дисципліни: Розробка ігрових застосунків на основі jMonkeyEngine 3

**Виконав:**

Студент групи 122-18-1

Мануйленко Іван Миколайович

**Перевірив:**

Реута О.В.

**Дніпро**

**2021**

Лабораторна робота №2

**Взаємодія із користувачем**

Мета роботи – ознайомитись з організацією взаємодії користувача з грою, навчитися реалізовувати цикл оновлення сцени і встановлювати реакції на дії користувача, розробити інтерактивну гру на основі персонажу, створеного в попередній роботі.

**Варіант 20**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Таблиця. Варіанти взаємодій **№** за  **списком** | **CA** | **C’A** | **CB** | **C’B** | **CC** | **C’C** | **OA** | **OB** | **OC** |
| **1** | Red | Orange | Blue | Cyan | Gray | White | OX | OX | OX |
| **2** | Red | Brown | Green | Cyan | Gray | Lightgray | OX | OX | OY |
| **3** | Red | Pink | Blue | Green | Gray | Darkgray | OX | OX | OZ |
| **4** | Red | Magenta | Blue | Cyan | Lightgray | White | OX | OY | OX |
| **5** | Orange | Brown | Green | Cyan | Darkgray | White | OX | OY | OY |
| **6** | Orange | Pink | Blue | Green | Lightgray | Darkgray | OX | OY | OZ |
| **7** | Orange | Magenta | Blue | Cyan | Gray | White | OX | OZ | OX |
| **8** | Brown | Pink | Green | Cyan | Gray | Lightgray | OX | OZ | OY |
| **9** | Brown | Magenta | Blue | Green | Gray | Darkgray | OX | OZ | OZ |
| **10** | Pink | Magenta | Blue | Cyan | Lightgray | White | OY | OX | OX |
| **11** | Red | Orange | Green | Cyan | Darkgray | White | OY | OX | OY |
| **12** | Red | Brown | Blue | Green | Lightgray | Darkgray | OY | OX | OZ |
| **13** | Red | Pink | Blue | Cyan | Gray | White | OY | OY | OX |
| **14** | Red | Magenta | Green | Cyan | Gray | Lightgray | OY | OY | OY |
| **15** | Orange | Brown | Blue | Green | Gray | Darkgray | OY | OY | OZ |
| **16** | Orange | Pink | Blue | Cyan | Lightgray | White | OY | OZ | OX |
| **17** | Orange | Magenta | Green | Cyan | Darkgray | White | OY | OZ | OY |
| **18** | Brown | Pink | Blue | Green | Lightgray | Darkgray | OY | OZ | OZ |
| **19** | Brown | Magenta | Blue | Cyan | Gray | White | OZ | OX | OX |
| **20** | Pink | Magenta | Green | Cyan | Gray | Lightgray | OZ | OX | OY |
| **21** | Red | Orange | Blue | Green | Gray | Darkgray | OZ | OX | OZ |
| **22** | Red | Brown | Blue | Cyan | Lightgray | White | OZ | OY | OX |
| **23** | Red | Pink | Green | Cyan | Darkgray | White | OZ | OY | OY |
| **24** | Red | Magenta | Blue | Green | Lightgray | Darkgray | OZ | OY | OZ |
| **25** | Orange | Brown | Blue | Cyan | Gray | White | OZ | OZ | OX |
| **26** | Orange | Pink | Green | Cyan | Gray | Lightgray | OZ | OZ | OY |
| **27** | Orange | Magenta | Blue | Green | Gray | Darkgray | OZ | OZ | OZ |
| **28** | Brown | Pink | Blue | Cyan | Lightgray | White | OX | OX | OX |
| **29** | Brown | Magenta | Green | Cyan | Darkgray | White | OX | OX | OY |
| **30** | Pink | Magenta | Blue | Green | Gray | White | OX | OX | OZ |
| **31** | Red | Orange | Blue | Cyan | Gray | Lightgray | OX | OY | OX |
| **32** | Red | Brown | Green | Cyan | Gray | Darkgray | OX | OY | OY |
| **33** | Red | Pink | Blue | Green | Lightgray | White | OX | OY | OZ |
| **34** | Red | Magenta | Blue | Cyan | Darkgray | White | OX | OZ | OX |
| **35** | Orange | Brown | Green | Cyan | Gray | White | OX | OZ | OY |

**Опис завдання**:

Реалізуйте наступну інтерактивну гру.

**Учасники гри:** гравець (користувач) і три персонажі A, B, C (фігури чоловічків в капелюшках з лабораторної роботи №1).

**Початок гри.** Після запуску гри персонажі розташовані на сцені в довільному порядку. Їх капелюшки жовтого кольору, а тіла кольорів CA, CB і CC, відповідно (всі позначені параметри наведені в таблиці в кінці інструкції).

**Сценарії гри.**

Сценарій 1**.** Гравець натискає клавіші 1, 2, 3 і відповідно персонажі A, B, C змінюють свої кольори. Наприклад, колір персонажу А — CA. Гравець натискає клавішу 1 послідовно три рази. Відповідно, колір персонажу А послідовно стає C’A, CA і C’A. Якщо користувач натискає клавішу 2, колір змінює персонаж В. Колір капелюхів завжди залишається незмінним. Сценарій 2**.** Гравець натискає клавіші F, G, H і відповідно персонажі A, B, C починають обертатися навколо заданих осей. Наприклад, персонаж А не обертається. Гравець натискає клавішу F послідовно три рази. Відповідно, персонаж А зразу обертається в додатному напрямку навколо заданої осі OA, потім — у від’ємному, а потім припиняє обертання. Якщо клавіша не натиснута, персонаж зберігає поточний стан (не обертається, або обертається в додатному або зворотному напрямку навколо своєї осі). Осі для персонажів A, B, C позначені в таблиці варіантів як OA, OB і ОС, відповідно.

Сценарій 3**.** Гравець натискає клавіші <, >, ? і відповідно персонажі A, B, C починають пульсувати (збільшуватись-зменшуватись, залишаючись в межах більше-менше в 1.5 рази, відносно первинного розміру). Наприклад, персонаж А має свій первинний розмір. Гравець натискає клавішу <. Персонаж А починає пульсувати: він збільшується, ставши в 1.5 разів більше, починає зменшуватись, ставши в 1.5 рази менше за свій оригінальний розмір, знов починає збільшуватись, і так далі. Коли гравець знову натискає клавішу <, персонаж А припиняє пульсацію, зберігаючи свій поточний розмір.

Сценарій 4**.** Гравець натискає клавішу Z і всі персонажі переходять в стан як на початку гри.

Сценарій 5**.** Гравець наводить перехрестя на капелюх будь-якого з персонажів, натискає ліву кнопку миші і капелюх зникає.

Посилання на віддалений репозиторій **Github** :

**Код:**

**Main.java**

public class Main extends SimpleApplication {

 private toGetCharacter character1;

private toGetCharacter character2;

private toGetCharacter character3;

boolean foRC1 = false;

boolean foRC2 = false;

boolean foRC3 = false;

ControlToRotate ctr1;

ControlToRotate ctr2;

ControlToRotate ctr3;

int status1 ;

int status2 ;

int status3 ;

boolean shiftIsPressed = false;

boolean check1 = false;

boolean check2 = false;

boolean check3 = false;

private ChangeSizeOfSpatial css1;

private ChangeSizeOfSpatial css2;

private ChangeSizeOfSpatial css3;

private Node nodes;

private ActionListener actionListener1;

  private ActionListener actionListener2;

   private ActionListener actionListener3;

     private ActionListener actionListener4;

    private ActionListener actionListener5;

    public static void main(String[] args) {

        Main app = new Main();

        app.start();

    }

//OZ OX OY

    @Override

    public void simpleInitApp()

    {

        initCharacters();

        initTriggers();

        initCrossHairs() ;

        nodes  = new Node("nodes");

        rootNode.attachChild(nodes);

        nodes.attachChild(character1.getCharacter());

        nodes.attachChild(character2.getCharacter());

        nodes.attachChild(character3.getCharacter());

        status1 = 1;

        status2 = 1;

        status3 = 1;

        ctr1 = null;

        ctr2 = null;

        ctr3 = null;

        css1 = null;

        css2 = null;

        css3 = null;

    }

    public void initCharacters()

    {

        Material matForHats = new Material(assetManager,"Common/MatDefs/Misc/Unshaded.j3md");

        matForHats.setColor("Color", ColorRGBA.Yellow);

        this.character1 = new toGetCharacter(new Vector3f(1,0,3),new Material(assetManager,"Common/MatDefs/Misc/Unshaded.j3md"),matForHats);

        this.character2 = new toGetCharacter(new Vector3f(3.8f,0,3),new Material(assetManager,"Common/MatDefs/Misc/Unshaded.j3md"),matForHats);

        this.character3 = new toGetCharacter(new Vector3f(6.6f,0,3),new Material(assetManager,"Common/MatDefs/Misc/Unshaded.j3md"),matForHats);

        this.character1.getMatForBody().setColor("Color", ColorRGBA.Pink);

        this.character2.getMatForBody().setColor("Color", ColorRGBA.Green);

        this.character3.getMatForBody().setColor("Color", ColorRGBA.Gray);

        character1.createABody();

        character2.createABody();

        character3.createABody();

    }

    //Magenta C`A Cyan C`B LightGray C`C

    private void  initTriggers()

    {

    inputManager.addMapping("ChangeColorCharacterOne", new KeyTrigger(KeyInput.KEY\_1));

    inputManager.addMapping("ChangeColorCharacterTwo", new KeyTrigger(KeyInput.KEY\_2));

    inputManager.addMapping("ChangeColorCharacterThree", new KeyTrigger(KeyInput.KEY\_3));

    inputManager.addMapping("Rotate1", new KeyTrigger(KeyInput.KEY\_F));

    inputManager.addMapping("Rotate2", new KeyTrigger(KeyInput.KEY\_G));

    inputManager.addMapping("Rotate3", new KeyTrigger(KeyInput.KEY\_H));

    inputManager.addMapping("SizeChange1", new KeyTrigger(KeyInput.KEY\_COMMA));

    inputManager.addMapping("SizeChange2", new KeyTrigger(KeyInput.KEY\_PERIOD));

    inputManager.addMapping("SizeChange3", new KeyTrigger(KeyInput.KEY\_SLASH));

    inputManager.addMapping("ToPress", new KeyTrigger(KeyInput.KEY\_LSHIFT));

     inputManager.addMapping("begin", new KeyTrigger(KeyInput.KEY\_Z));

     inputManager.addMapping("clickOnHat",new MouseButtonTrigger(MouseInput.BUTTON\_LEFT));

   actionListener1 = new ActionListener() {

    public void onAction(String name, boolean keyPressed, float tpf) {

      if (name.equals("ChangeColorCharacterOne") && keyPressed) {

          if(!foRC1)

          {

           character1.getMatForBody().setColor("Color", ColorRGBA.Magenta);

           foRC1 = true;

          }

           else

          {

               character1.getMatForBody().setColor("Color", ColorRGBA.Pink);

               foRC1 = false;

          }

      }

     else if (name.equals("ChangeColorCharacterTwo") && keyPressed) {

           if(!foRC2)

          {

           character2.getMatForBody().setColor("Color", ColorRGBA.Cyan);

           foRC2 = true;

          }

           else

          {

               character2.getMatForBody().setColor("Color", ColorRGBA.Green);

               foRC2 = false;

          }

      }

          else if (name.equals("ChangeColorCharacterThree") && keyPressed) {

            if(!foRC3)

          {

           character3.getMatForBody().setColor("Color", ColorRGBA.LightGray);

           foRC3 = true;

          }

           else

          {

               character3.getMatForBody().setColor("Color", ColorRGBA.Gray);

               foRC3 = false;

          }

      }

    }

  };

      actionListener2 = new ActionListener() {

    public void onAction(String name, boolean keyPressed, float tpf) {

      if (name.equals("Rotate1") && keyPressed) {

          if(status1 == 1){

          if(ctr1 == null)

          {

              ctr1 = new ControlToRotate();

          }

          ctr1.setAngle(new Vector3f(0,0,FastMath.DEG\_TO\_RAD\*10));

          ctr1.setSpeed(40);

          character1.getCharacter().addControl(ctr1);

          status1 = 2;

          }

          else if (status1 ==2)

          {

              if(ctr1 == null)

              {

                   ctr1 = new ControlToRotate();

              }

               ctr1.setSpeed(40);

               ctr1.setAngle(new Vector3f(0,0,-FastMath.DEG\_TO\_RAD\*10));

              status1 = 3;

          }

              else

          {

              character1.getCharacter().removeControl(ctr1);

              ctr1 = null;

              status1 = 1;

          }

      }

     else if (name.equals("Rotate2") && keyPressed) {

              if(status2 == 1){

          if(ctr2 == null)

          {

              ctr2 = new ControlToRotate();

          }

          ctr2.setAngle(new Vector3f(FastMath.DEG\_TO\_RAD\*10,0,0));

          ctr2.setSpeed(40);

          character2.getCharacter().addControl(ctr2);

          status2 = 2;

          }

          else if (status2 ==2)

          {

              if(ctr2 == null)

              {

                   ctr2 = new ControlToRotate();

              }

               ctr2.setSpeed(40);

               ctr2.setAngle(new Vector3f(-FastMath.DEG\_TO\_RAD\*10,0,0));

               status2 = 3;

          }

              else

          {

              character2.getCharacter().removeControl(ctr2);

              ctr2 = null;

              status2 = 1;

          }

      }

          else if (name.equals("Rotate3") && keyPressed) {

                    if(status3 == 1){

          if(ctr3 == null)

          {

              ctr3 = new ControlToRotate();

          }

          ctr3.setAngle(new Vector3f(0,FastMath.DEG\_TO\_RAD\*10,0));

          ctr3.setSpeed(40);

          character3.getCharacter().addControl(ctr3);

          status3 = 2;

          }

          else if (status3 ==2)

          {

              if(ctr3 == null)

              {

                   ctr3 = new ControlToRotate();

              }

               ctr3.setSpeed(40);

               ctr3.setAngle(new Vector3f(0,-FastMath.DEG\_TO\_RAD\*10,0));

               status3 = 3;

          }

              else

          {

              character3.getCharacter().removeControl(ctr3);

              ctr3 = null;

              status3 = 1;

          }

      }

    }

  };

     actionListener3 = new ActionListener() {

    public void onAction(String name, boolean keyPressed, float tpf)

    {

        if(name.equals("ToPress")&&keyPressed){

            shiftIsPressed = !shiftIsPressed;

        }

        else if(name.equals("ToPress")&&!keyPressed)

             {

                 shiftIsPressed = !shiftIsPressed;

             }

        else if(name.equals("SizeChange1")&&!keyPressed && shiftIsPressed)

        {

            if(css1 ==  null)

            {

                css1 = new ChangeSizeOfSpatial();

                css1.setSpeed(10);

            }

            if(!check1){

            character1.getCharacter().addControl(css1);

            check1 = true;

            }

            else

            {

                character1.getCharacter().removeControl(css1);

                check1 = false;

            }

        }

        else if(name.equals("SizeChange2")&&!keyPressed&&shiftIsPressed)

             {

            if(css2 ==  null)

            {

                css2 = new ChangeSizeOfSpatial();

                css2.setSpeed(10);

            }

            if(!check2){

            character2.getCharacter().addControl(css2);

            check2 = true;

            }

            else

            {

                character2.getCharacter().removeControl(css2);

                check2 = false;

            }

             }

        else if (name.equals("SizeChange3")&&!keyPressed&&shiftIsPressed)

             {

                     if(css3 ==  null)

            {

                css3 = new ChangeSizeOfSpatial();

                css3.setSpeed(10);

            }

            if(!check3){

            character3.getCharacter().addControl(css3);

            check3 = true;

            }

            else

            {

                character3.getCharacter().removeControl(css3);

                check3 = false;

            }

             }

    }

  };

    actionListener4 = new ActionListener() {

            public void onAction(String name, boolean keyPressed, float tpf)

            {

                if(name.equals("begin")&&keyPressed)

                {

                  character1.getCharacter().removeControl(ctr1);

                  character1.getCharacter().removeControl(css1);

                  character2.getCharacter().removeControl(ctr2);

                  character2.getCharacter().removeControl(css2);

                  character3.getCharacter().removeControl(ctr3);

                  character3.getCharacter().removeControl(css3);

                   rootNode.detachChild(nodes);

                   nodes.detachAllChildren();

                   initCharacters();

                   rootNode.attachChild(nodes);

                       nodes.attachChild(character1.getCharacter());

                        nodes.attachChild(character2.getCharacter());

                       nodes.attachChild(character3.getCharacter());

        status1 = 1;

        status2 = 1;

        status3 = 1;

        ctr1 = null;

        ctr2 = null;

        ctr3 = null;

        css1 = null;

        css2 = null;

        css3 = null;

                }

    }

  };

   actionListener5 = new ActionListener() {

            public void onAction(String name, boolean keyPressed, float tpf)

            {

                if(name.equals("clickOnHat")&&keyPressed)

                {

        CollisionResults results = new CollisionResults();

        Ray ray = new Ray(cam.getLocation(), cam.getDirection());

        nodes.collideWith(ray, results);

        if (results.size() > 0) {

          CollisionResult closest = results.getClosestCollision();

           Node hat = closest.getGeometry().getParent();

           hat.getParent().detachChild(hat);

        }

                }

    }

  };

     inputManager.addListener(actionListener1, new String[]{"ChangeColorCharacterOne","ChangeColorCharacterTwo","ChangeColorCharacterThree"});

     inputManager.addListener(actionListener2, new String[]{"Rotate1","Rotate2","Rotate3"});

     inputManager.addListener(actionListener3, new String[]{"ToPress","SizeChange1","SizeChange2","SizeChange3"});

     inputManager.addListener(actionListener4, "begin");

       inputManager.addListener(actionListener5, "clickOnHat");

    }

     protected void initCrossHairs()

    {

    setDisplayStatView(false);

    guiFont = assetManager.loadFont("Interface/Fonts/Default.fnt");

    BitmapText ch = new BitmapText(guiFont, false);

    ch.setSize(guiFont.getCharSet().getRenderedSize() \* 2);

    ch.setText("+"); // crosshairs

    ch.setLocalTranslation( // center

      settings.getWidth() / 2 - ch.getLineWidth()/2,

      settings.getHeight() / 2 + ch.getLineHeight()/2, 0);

    guiNode.attachChild(ch);

  }

    @Override

    public void simpleUpdate(float tpf) {

    }

    @Override

    public void simpleRender(RenderManager rm) {

    }

}

**toGetCharacter.java**

class toGetCharacter {

  Material matForBodyElements ;

  Material mathForHatElements ;

  Vector3f originPos;

 private Node bodyelements;

  private Node hat;

  private Node bodyParts;

    public toGetCharacter(Vector3f originPos,Material mfbe,Material mfhe)

    {

        this.matForBodyElements = mfbe;

        this.mathForHatElements = mfhe;

        this.originPos = originPos;

        this.bodyParts = new Node("Bodyparts");

        this.bodyelements = new Node("BodyElements");

        this.hat = new Node("hat");

    }

      public Material getMatForBody()

  {

      return this.matForBodyElements;

  }

      public Material getMatForHat()

      {

      return this.mathForHatElements;

      }

    public Node getBody()

    {

        return this.bodyelements;

    }

    public Node getHat()

    {

        return this.hat;

    }

    public void createABody()

    {

    this.bodyParts.setLocalTranslation(this.originPos);

    Geometry body = new Geometry("body",new Cylinder(10,10,0.5f,1f,true));

     body.rotate(FastMath.DEG\_TO\_RAD\*90, 0, 0);

     Geometry head = new Geometry("head",new Sphere(10,10,0.35f));

     head.move(0,0.85f,0);

    Geometry rightArm = new Geometry("rightarm",new Cylinder(10,10,0.2f,0.7f,true));

    rightArm.move(0.9f,0,0);

    rightArm.rotate(FastMath.DEG\_TO\_RAD\*90, 0, FastMath.DEG\_TO\_RAD\*45);

    Geometry leftArm = new Geometry("leftarm",new Cylinder(10,10,0.2f,0.7f,true));

         leftArm.move(-0.9f,0,0);

         leftArm.rotate(FastMath.DEG\_TO\_RAD\*90, 0, -FastMath.DEG\_TO\_RAD\*45);

    Geometry leftLeg = new Geometry("leftleg",new Cylinder(10,10,0.28f,0.8f,true));

    leftLeg.move(-0.35f,-1,0);

       leftLeg.rotate(FastMath.DEG\_TO\_RAD\*90,0,0);

    Geometry rightLeg = new Geometry("rightleg",new Cylinder(10,10,0.28f,0.8f,true));

     rightLeg.move(0.35f,-1,0);

     rightLeg.rotate(FastMath.DEG\_TO\_RAD\*90,0,0);

    this.bodyelements.attachChild(body);

      this.bodyelements.attachChild(head);

        this.bodyelements.attachChild(leftArm);

         this.bodyelements.attachChild(rightArm);

            this.bodyelements.attachChild(leftLeg);

              this.bodyelements.attachChild(rightLeg);

           List<Spatial> bparts =   this.bodyelements.getChildren();

           for(Spatial part : bparts)

           {

                 if(part instanceof Geometry)

                 {

                     ((Geometry)part).setMaterial(this.matForBodyElements);

                 }

           }

    Geometry headGear = new Geometry("headgear",new Torus(10,10,0.15f,0.4f));

    Geometry headCylinder = new Geometry("headcylinder",new Cylinder(10,10,0.2f,0.5f,true));

    this.hat.attachChild(headGear);

    this.hat.attachChild(headCylinder);

    this.hat.rotate(-FastMath.DEG\_TO\_RAD\*90,0,0);

    this.hat.move(0,1.3f,0);

    this.hat.setMaterial(this.mathForHatElements);

    headCylinder.move(0,0,0.4f);

    this.bodyParts.attachChild( this.bodyelements);

    this.bodyParts.attachChild( this.hat);

    }

    public Node  getCharacter()

    {

    return this.bodyParts;

    }

}

**ControlToRotate.java**

public class ControlToRotate extends AbstractControl {

    private Vector3f angle;

    private float speed;

      public ControlToRotate(){}

        public ControlToRotate(Vector3f angleToRotate,float spd){

            angle = angleToRotate;

            speed = spd;

        }

      public void setAngle(Vector3f angleToRotate)

      {

          angle = angleToRotate;

      }

      public Vector3f getAngle()

      {

          return angle;

      }

      public void setSpeed(float spd)

      {

          speed = spd;

      }

      public float getSpeed (float spd)

      {

          return speed;

      }

         @Override

    protected void controlUpdate(float tpf){

        if(spatial != null) {

            spatial.rotate(angle.x\*tpf\*speed,angle.y\*tpf\*speed,angle.z\*tpf\*speed);

        }

    }

  @Override

    public void setSpatial(Spatial spatial) {

        super.setSpatial(spatial);

        /\* Example:

        if (spatial != null){

            // initialize

        }else{

            // cleanup

        }

        \*/

    }

    @Override

    public ControlToRotate cloneForSpatial(Spatial spatial)

    {

        final ControlToRotate control = new ControlToRotate(angle,speed);

        control.setSpatial(spatial);

        return control;

    }

     @Override

    protected void controlRender(RenderManager rm, ViewPort vp){

    }

}

**ChangeSizeOfSpatial.java**

public class ChangeSizeOfSpatial extends AbstractControl {

    private Vector3f localScale;

     private Vector3f minScale;

     private Vector3f maxScale;

    private float speed;

    private boolean ud;

    private Vector3f addUpTo;

      public ChangeSizeOfSpatial(){}

      public void setLocalScale(Vector3f local)

      {

          localScale = local;

      }

      public Vector3f getLocalScale()

      {

          return localScale;

      }

      public void setSpeed(float spd)

      {

          speed = spd;

      }

      public float getSpeed (float spd)

      {

          return speed;

      }

         @Override

    protected void controlUpdate(float tpf){

        if(spatial != null) {

            System.out.println(speed+" "+ud+" "+localScale+" "+maxScale+" "+minScale+" "+addUpTo);

            if(ud)

            {

                localScale=localScale.add(addUpTo);

                spatial.setLocalScale(localScale);

                if(vectorEquality(localScale,maxScale))ud = false;

            }

            else

            {

               localScale = localScale.subtract(addUpTo);

                spatial.setLocalScale(localScale);

                if(vectorEquality(minScale,localScale))ud = true;

            }

        }

    }

    public boolean vectorEquality(Vector3f fv,Vector3f sv)

    {

        return (fv.x)>(sv.x)?(fv.y)>(sv.y)?(fv.z)>(sv.z):false:false;

    }

  @Override

    public void setSpatial(Spatial spatial) {

        super.setSpatial(spatial);

        if(spatial != null)

        {

           localScale = spatial.getLocalScale();

           minScale = localScale.clone();

           maxScale = minScale.clone().mult(1.5f);

           addUpTo = maxScale.clone().subtract(minScale);

           addUpTo = addUpTo.divide(speed);

           ud = true;

        }

        else

        {

            localScale = null;

            maxScale = null;

            minScale = null;

            addUpTo = null;

        }

        /\* Example:

        if (spatial != null){

            // initialize

        }else{

            // cleanup

        }

        \*/

    }

    @Override

    public ControlToRotate cloneForSpatial(Spatial spatial)

    {

        final ControlToRotate control = new ControlToRotate();

        control.setSpatial(spatial);

        return control;

    }

     @Override

    protected void controlRender(RenderManager rm, ViewPort vp){

    }

}

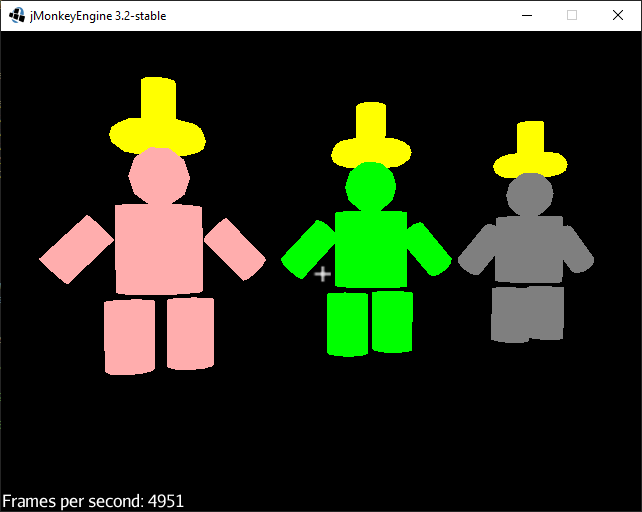


Рис.1 Початок гри



Рис.2 Зміна кольору

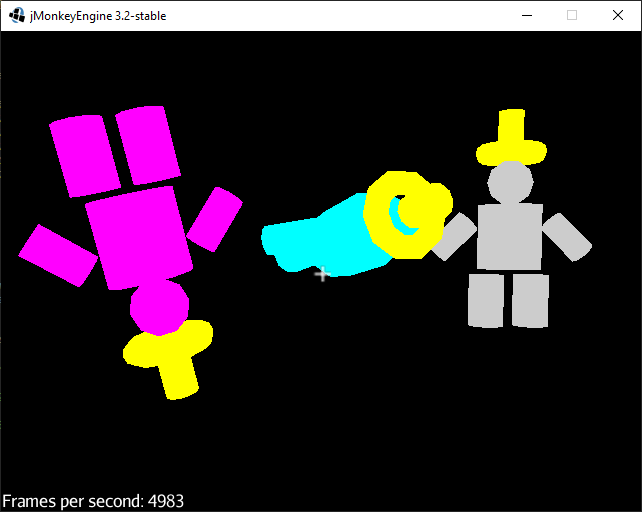


Рис.3 Обертання навколо заданих осей у обох напрямках

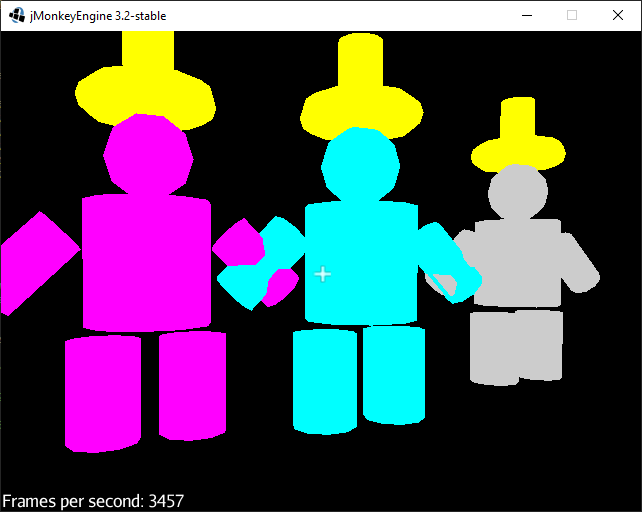


Рис.4 Пульсування розмірів персонажів

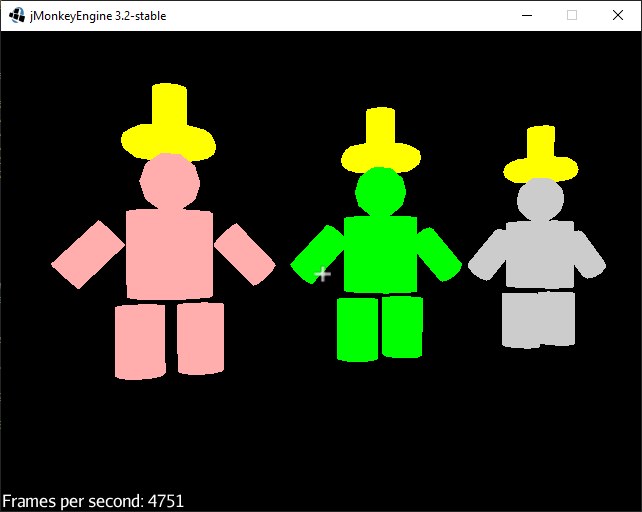


Рис.5 Перехід до початкового стану гри

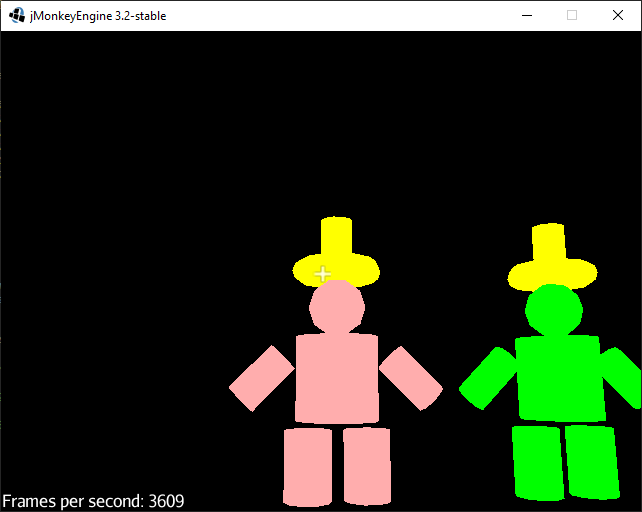


Рис.6 До натискання на ліву кнопку миші

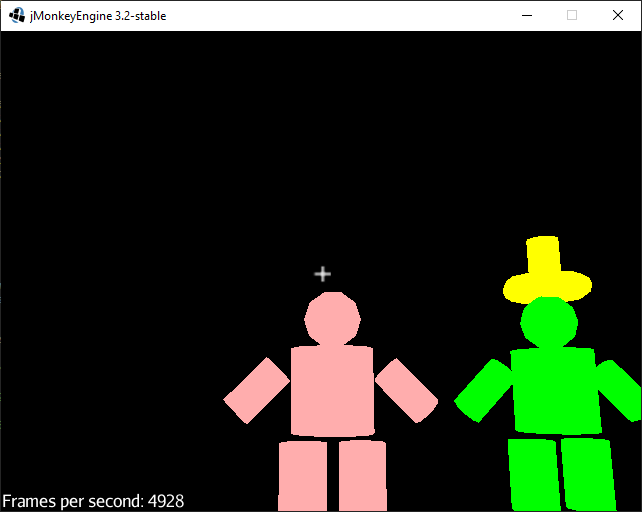


Рис.7 Зникнення капелюха персонажу при натисканні лівої кнопки миші на нього

Висновок: я дізнався про основні можливості трансформації, дізнався про можливість взаємодії з користувачем за допомогою трігерів, та також навчився отримувати об’єкт, на який натискає користувач.