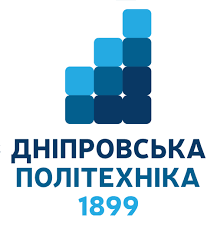
Міністерство освіти і науки України.

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 Павло Миколайович

**Перевірив:**

Реута О.В.

**Дніпро**

**2021**

Лабораторна робота №2

Варіант 21

**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.

Варіанти завдань

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

**Продовження таблиці варіантів завдань**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** :

<https://github.com/Dozez13/JmonkeyEngine>

**Код**

**Person class**

/\*

 \* To change this license header, choose License Headers in Project Properties.

 \* To change this template file, choose Tools | Templates

 \* and open the template in the editor.

 \*/

package mygame;

import com.jme3.material.Material;

import com.jme3.math.FastMath;

import com.jme3.math.Quaternion;

import com.jme3.math.Vector3f;

import com.jme3.scene.Geometry;

import com.jme3.scene.Node;

import com.jme3.scene.Spatial;

import com.jme3.scene.shape.Box;

import com.jme3.scene.shape.Cylinder;

import com.jme3.scene.shape.Sphere;

import com.jme3.scene.shape.Torus;

import java.util.List;

/\*\*

 \*

 \* @author Pavel

 \*/

public class Person implements Cloneable{

     private static int count=0;

     private  Material body;

     private  Material hat;

     private Node all;

    public Person(Material body,Material hat){

      this.body = body;

      this.hat = hat;

    }

    public Node getAll(){

        return this.all;

    }

    public Material getBody(){

        return this.body;

    }

    public void setAll(Node all){

        this.all = all;

    }

    public Node personToNode(){

        Geometry bodys = new Geometry("body",new Sphere(100,100,0.8f));

        Geometry head = new Geometry("head",new Box(0.5f,0.5f,0.5f));

        Geometry leftHand = new Geometry("leftHand",new Box(0.5f,0.5f,0.5f));

        Geometry rightHand = new Geometry("rightHand",new Box(0.5f,0.5f,0.5f));

        Geometry leftLeg = new Geometry("leftLeg",new Box(0.5f,0.6f,0.5f));

        Geometry rightLeg = new Geometry("rightLeg",new Box(0.5f,0.6f,0.5f));

        Geometry brim = new Geometry("crown",new Torus(20,20,0.1f,0.5f));

        Geometry crown = new Geometry("brim",new Cylinder(20,20,0.4f,0.6f,true));

        Node hatElements = new Node("hat");

        hatElements.attachChild(crown);

        hatElements.attachChild(brim);

        crown.move(0,0,0.3f);

        hatElements.rotate(-FastMath.PI/2,0,0);

        hatElements.move(0,2.5f,0);

        Node person = new Node("Body");

        person.attachChild(leftLeg);

        person.attachChild(rightLeg);

        person.attachChild(leftHand);

        person.attachChild(rightHand);

        person.attachChild(bodys);

        person.attachChild(head);

        leftLeg.move(-0.7f,-1.2f,0);

        rightLeg.move(0.7f,-1.2f,0);

        leftHand.move(-1f,0.6f,0);

        rightHand.move(1f,0.6f,0);

        head.move(0,1.8f,0);

        List<Spatial> spatials = person.getChildren();

        for(Spatial sp:spatials){

            if(sp instanceof Geometry){

                ((Geometry)sp).setMaterial(this.body);

            }

        }

          spatials = hatElements.getChildren();

        for(Spatial sp:spatials){

            if(sp instanceof Geometry){

                ((Geometry)sp).setMaterial(this.hat);

            }

        }

       Quaternion quaternion = new Quaternion();

        quaternion.fromAngleAxis(FastMath.DEG\_TO\_RAD+count\*15, Vector3f.UNIT\_Y);

        all = new Node("all");

        all.attachChild(person);

        all.attachChild(hatElements);

        all.move(count\*5,0,count);

        all.rotate(quaternion);

        count++;

        return all;

    }

    @Override

    protected Object clone()throws CloneNotSupportedException

    {

        Person person = (Person)super.clone();

        person.all = (Node)person.all.clone();

        person.body = (Material)person.body.clone();

        person.hat = (Material)person.hat.clone();

        return person;

    }

}

**Task2 class**

package mygame;

import com.jme3.app.SimpleApplication;

import com.jme3.collision.CollisionResults;

import com.jme3.font.BitmapText;

import com.jme3.input.controls.ActionListener;

import com.jme3.input.controls.AnalogListener;

import com.jme3.material.Material;

import com.jme3.math.ColorRGBA;

import com.jme3.math.FastMath;

import com.jme3.math.Ray;

import com.jme3.math.Vector3f;

import com.jme3.renderer.RenderManager;

import com.jme3.scene.Node;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

 \* This is the Main Class of your Game. You should only do initialization here.

 \* Move your Logic into AppStates or Controls

 \* @author Pavel

 \*/

public class Task2 extends SimpleApplication {

    private Person firstPerson,secondPerson,thirdPerson;

    private Person firstDefault,secondDefault,thirdDefault;

    public  Material FIRST,SECOND,THIRD;

    private boolean lShiftPressed;

    private boolean[] toNotIncrease = new boolean[3];

    private float[] scaleMeasure = new float[3];

    private boolean[] color = new boolean[3];

    private boolean[] rotate = new boolean[3];

    private byte a=-1;

    ControlToRotate ctr1;

ControlToRotate ctr2;

ControlToRotate ctr3;

int status1 =1;

int status2 =1;

int status3 =1;

    private final ActionListener changeDirection = new ActionListener(){

        @Override

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

            if(!isPressed){

            switch(name){

                case UserInput.MAPPING\_ROTATE\_OF\_FIRST:{

                      rotate[0] = !rotate[0];

                    break;

                }

                case UserInput.MAPPING\_ROTATE\_OF\_SECOND:{

                    rotate[1] = !rotate[1];

                    break;

                }

                case UserInput.MAPPING\_ROTATE\_OF\_THIRD:{

                    rotate[2] = !rotate[2];

                    break;

                }

            }

            }

        }

    };

    private final ActionListener actionListener = new ActionListener() {

        @Override

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

                if(keyPressed){

                switch(name){

                case UserInput.MAPPING\_COLOR\_OF\_FIRST:{

                       if(!color[0]){

                       firstPerson.getBody().setColor("Color",ColorRGBA.Orange);

                       color[0]= true;

                       }else{

                           firstPerson.getBody().setColor("Color", ColorRGBA.Red);

                           color[0]= false;

                       }

                    break;

                }

                 case UserInput.MAPPING\_COLOR\_OF\_SECOND:{

                       if(!color[1]){

                       secondPerson.getBody().setColor("Color",ColorRGBA.Green);

                       color[1]= true;

                       }else{

                           secondPerson.getBody().setColor("Color", ColorRGBA.Blue);

                           color[1]= false;

                       }

                    break;

                }

                  case UserInput.MAPPING\_COLOR\_OF\_THIRD:{

                        if(!color[2]){

                       thirdPerson.getBody().setColor("Color",ColorRGBA.DarkGray);

                         color[2]= true;

                       }else{

                           thirdPerson.getBody().setColor("Color", ColorRGBA.Gray);

                          color[2]= false;

                       }

                   break;

                }

            }

         }

        }

    };

   private final AnalogListener analog = new AnalogListener(){

        @Override

        public void onAnalog(String name, float value, float tpf) {

                switch(name){

                case UserInput.MAPPING\_ROTATE\_OF\_FIRST:{

                        if(!rotate[0]){

                            firstPerson.getAll().rotate(0,0,FastMath.DEG\_TO\_RAD\*2);

                        }else{

                            firstPerson.getAll().rotate(0,0,-FastMath.DEG\_TO\_RAD\*2);

                        }

                       a=0;

                    break;

                }

                 case UserInput.MAPPING\_ROTATE\_OF\_SECOND:{

                     if(!rotate[1]){

                         secondPerson.getAll().rotate(FastMath.DEG\_TO\_RAD\*2,0,0);

                     }else{

                         secondPerson.getAll().rotate(-FastMath.DEG\_TO\_RAD\*2,0,0);

                     }

                      a=1;

                    break;

                }

                  case UserInput.MAPPING\_ROTATE\_OF\_THIRD:{

                      if(!rotate[2]){

                          thirdPerson.getAll().rotate(0,0,FastMath.DEG\_TO\_RAD\*2);

                      }else{

                          thirdPerson.getAll().rotate(0,0,-FastMath.DEG\_TO\_RAD\*2);

                      }

                      a=2;

                   break;

                }

            }

         }

   };

   private final ActionListener toBegin = new ActionListener(){

        @Override

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

            if(!isPressed){

                    firstPerson.getAll().removeControl(ctr1);

                     secondPerson.getAll().removeControl(ctr2);

                      thirdPerson.getAll().removeControl(ctr3);

                    firstPerson.getBody().setColor("Color", ColorRGBA.Red);

                    secondPerson.getBody().setColor("Color", ColorRGBA.Blue);

                    thirdPerson.getBody().setColor("Color", ColorRGBA.Gray);

                try {

                    firstPerson.getAll().attachChild(((Person)firstDefault.clone()).getAll().getChild("hat"));

                    secondPerson.getAll().attachChild(((Person)secondDefault.clone()).getAll().getChild("hat"));

                    thirdPerson.getAll().attachChild(((Person)thirdDefault.clone()).getAll().getChild("hat"));

                    firstPerson.getAll().setLocalRotation(((Person)firstDefault.clone()).getAll().getLocalRotation());

                    secondPerson.getAll().setLocalRotation(((Person)secondDefault.clone()).getAll().getLocalRotation());

                    thirdPerson.getAll().setLocalRotation(((Person)thirdDefault.clone()).getAll().getLocalRotation());

                    firstPerson.getAll().setLocalScale(((Person)firstDefault.clone()).getAll().getLocalScale());

                    secondPerson.getAll().setLocalScale(((Person)secondDefault.clone()).getAll().getLocalScale());

                    thirdPerson.getAll().setLocalScale(((Person)thirdDefault.clone()).getAll().getLocalScale());

                } catch (CloneNotSupportedException ex) {

                    Logger.getLogger(Task2.class.getName()).log(Level.SEVERE, null, ex);

                }

     }

     }

   };

   private final ActionListener actionListener2 = new ActionListener() {

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

      if (name.equals(UserInput.MAPPING\_ROTATE\_OF\_FIRST) && keyPressed) {

          if(status1 == 1){

          if(ctr1 == null)

          {

              ctr1 = new ControlToRotate();

          }

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

          ctr1.setSpeed(40);

          firstPerson.getAll().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

          {

               firstPerson.getAll().removeControl(ctr1);

              ctr1 = null;

              status1 = 1;

          }

      }

     else if (name.equals(UserInput.MAPPING\_ROTATE\_OF\_SECOND) && keyPressed) {

              if(status2 == 1){

          if(ctr2 == null)

          {

              ctr2 = new ControlToRotate();

          }

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

          ctr2.setSpeed(40);

           secondPerson.getAll().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

          {

              secondPerson.getAll().removeControl(ctr2);

              ctr2 = null;

              status2 = 1;

          }

      }

          else if (name.equals(UserInput.MAPPING\_ROTATE\_OF\_THIRD) && keyPressed) {

                    if(status3 == 1){

          if(ctr3 == null)

          {

              ctr3 = new ControlToRotate();

          }

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

          ctr3.setSpeed(40);

           thirdPerson.getAll().addControl(ctr3);

          status3 = 2;

          }

          else if (status3 ==2)

          {

              if(ctr3 == null)

              {

                   ctr3 = new ControlToRotate();

              }

               ctr3.setSpeed(40);

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

               status3 = 3;

          }

              else

          {

              thirdPerson.getAll().removeControl(ctr3);

              ctr3 = null;

              status3 = 1;

          }

      }

    }

  };

    private final ActionListener leftShift = new ActionListener(){

        @Override

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

            lShiftPressed = isPressed;

        }

    };

     private final ActionListener ScaleElement = new ActionListener(){

        @Override

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

            if(isPressed&&lShiftPressed){

                switch(name){

                    case UserInput.MAPPING\_COMMA:{

                  if(scaleMeasure[1]<1.5f&&!toNotIncrease[1]){

                  firstPerson.getAll().setLocalScale(scaleMeasure[1]);

                  scaleMeasure[1]+=0.05f;

              }

              else if(scaleMeasure[1]>0.66f){

                  toNotIncrease[1] = true;

                  scaleMeasure[1]-=0.05f;

                  firstPerson.getAll().setLocalScale(scaleMeasure[1]);

              }else{

                  toNotIncrease[1] = false;

              }

                        break;

                    }

                    case UserInput.MAPPING\_POINT:{

                   if(scaleMeasure[2]<1.5f&&!toNotIncrease[2]){

                  secondPerson.getAll().setLocalScale(scaleMeasure[2]);

                  scaleMeasure[2]+=0.05f;

              }

              else if(scaleMeasure[2]>0.66f){

                  toNotIncrease[2] = true;

                  scaleMeasure[2]-=0.05f;

                  secondPerson.getAll().setLocalScale(scaleMeasure[2]);

              }else{

                  toNotIncrease[2] = false;

              }

                       break;

                    }

                    case UserInput.MAPPING\_SLASH:{

                if(scaleMeasure[0]<1.5f&&!toNotIncrease[0]){

                  thirdPerson.getAll().setLocalScale(scaleMeasure[0]);

                  scaleMeasure[0]+=0.05f;

              }

              else if(scaleMeasure[0]>0.66f){

                  toNotIncrease[0] = true;

                  scaleMeasure[0]-=0.05f;

                  thirdPerson.getAll().setLocalScale(scaleMeasure[0]);

              }else{

                  toNotIncrease[0] = false;

              }

                        break;

                    }

                }

            }

        }

    };

    public static void main(String[] args) {

        Task2 app = new Task2();

        app.start();

    }

    public void initInstances(){

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

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

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

      this.FIRST.setColor("Color", ColorRGBA.Red);

      this.SECOND.setColor("Color", ColorRGBA.Blue);

      this.THIRD.setColor("Color", ColorRGBA.Gray);

      this.color[0] = false;

      this.color[1] = false;

      this.color[2] = false;

       this.rotate[0] = false;

      this.rotate[1] = false;

      this.rotate[2] = false;

      this.toNotIncrease[0] = false;

      this.toNotIncrease[1] = false;

      this.toNotIncrease[2] = false;

      this.scaleMeasure[0] = 1;

      this.scaleMeasure[1] = 1;

      this.scaleMeasure[2] = 1;

    }

    private void setupKeys(){

        inputManager.addMapping(UserInput.MAPPING\_COLOR\_OF\_FIRST, UserInput.TRIGGER\_COLOR1);

        inputManager.addMapping(UserInput.MAPPING\_COLOR\_OF\_SECOND, UserInput.TRIGGER\_COLOR2);

        inputManager.addMapping(UserInput.MAPPING\_COLOR\_OF\_THIRD, UserInput.TRIGGER\_COLOR3);

        inputManager.addMapping(UserInput.MAPPING\_ROTATE\_OF\_FIRST, UserInput.ROTATE\_FIRST);

        inputManager.addMapping(UserInput.MAPPING\_ROTATE\_OF\_SECOND, UserInput.ROTATE\_SECOND);

        inputManager.addMapping(UserInput.MAPPING\_ROTATE\_OF\_THIRD, UserInput.ROTATE\_THIRD);

        inputManager.addMapping(UserInput.MAPPING\_BEGIN,UserInput.TO\_BEGIN);

        inputManager.addMapping(UserInput.MAPPING\_REMOVING,UserInput.REMOVING\_HATS);

        inputManager.addMapping(UserInput.MAPPING\_SHIFT,UserInput.LEFT\_SHIFT);

        inputManager.addMapping(UserInput.MAPPING\_SLASH,UserInput.SLASH\_KEY);

        inputManager.addMapping(UserInput.MAPPING\_COMMA,UserInput.COMMA);

        inputManager.addMapping(UserInput.MAPPING\_POINT,UserInput.POINT);

        inputManager.addListener(ScaleElement, UserInput.MAPPING\_SLASH,UserInput.MAPPING\_COMMA,UserInput.MAPPING\_POINT);

        inputManager.addListener(leftShift, UserInput.MAPPING\_SHIFT);

        inputManager.addListener(removingHat,UserInput.MAPPING\_REMOVING);

        inputManager.addListener(actionListener,UserInput.MAPPING\_REMOVING);

        inputManager.addListener(toBegin, UserInput.MAPPING\_BEGIN);

        inputManager.addListener(actionListener, UserInput.MAPPING\_COLOR\_OF\_FIRST,UserInput.MAPPING\_COLOR\_OF\_SECOND,UserInput.MAPPING\_COLOR\_OF\_THIRD);

        inputManager.addListener(actionListener2, UserInput.MAPPING\_ROTATE\_OF\_FIRST,UserInput.MAPPING\_ROTATE\_OF\_SECOND,UserInput.MAPPING\_ROTATE\_OF\_THIRD);

    }

     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);

  }

    private final ActionListener removingHat = new ActionListener() {

    @Override

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

         CollisionResults results = new CollisionResults();

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

         rootNode.collideWith(ray, results);

         for (int i = 0; i < results.size(); i++) {

           String target = results.getCollision(i).getGeometry().getName();

           if("crown".equals(target)||"brim".equals(target)){

                results.getCollision(i).getGeometry().getParent().removeFromParent();

           }

         }

    }

  };

    @Override

    public void simpleInitApp()  {

        initCrossHairs();

        this.setupKeys();

        this.initInstances();

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

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

        firstPerson = new Person(FIRST,mat.clone());

        secondPerson = new Person(SECOND,mat.clone());

        thirdPerson = new Person(THIRD,mat.clone());

        Node node1 = firstPerson.personToNode();

        Node node2 = secondPerson.personToNode();

        Node node3 = thirdPerson.personToNode();

          try{

            this.firstDefault =(Person)firstPerson.clone();

            this.secondDefault =(Person) secondPerson.clone();

            this.thirdDefault = (Person)thirdPerson.clone();

        }catch(CloneNotSupportedException e){

            System.err.print(e.getMessage());

        }

        rootNode.attachChild(node1);

        rootNode.attachChild(node2);

        rootNode.attachChild(node3);

    }

    @Override

    public void simpleUpdate(float tpf) {

    }

    @Override

    public void simpleRender(RenderManager rm) {

        //TODO: add render code

    }

}

**UserInput class**

/\*

 \* To change this license header, choose License Headers in Project Properties.

 \* To change this template file, choose Tools | Templates

 \* and open the template in the editor.

 \*/

package mygame;

import com.jme3.input.\*;

import com.jme3.input.controls.\*;

/\*\*

 \*

 \* @author Pavel

 \*/

public class UserInput {

 public final static Trigger TRIGGER\_COLOR1 = new KeyTrigger(KeyInput.KEY\_1);

 public final static Trigger TRIGGER\_COLOR2 = new KeyTrigger(KeyInput.KEY\_2);

 public final static Trigger TRIGGER\_COLOR3 = new KeyTrigger(KeyInput.KEY\_3);

 public final static Trigger ROTATE\_FIRST = new KeyTrigger(KeyInput.KEY\_F);

 public final static Trigger ROTATE\_SECOND = new KeyTrigger(KeyInput.KEY\_G);

 public final static Trigger ROTATE\_THIRD = new KeyTrigger(KeyInput.KEY\_H);

 public final static Trigger TO\_BEGIN = new KeyTrigger(KeyInput.KEY\_Z);

 public final static Trigger REMOVING\_HATS = new MouseButtonTrigger(MouseInput.BUTTON\_LEFT);

 public final static Trigger SLASH\_KEY = new KeyTrigger(KeyInput.KEY\_SLASH);

 public final static Trigger COMMA = new KeyTrigger(KeyInput.KEY\_COMMA);

 public final static Trigger POINT = new KeyTrigger(KeyInput.KEY\_PERIOD);

 public final static Trigger LEFT\_SHIFT = new KeyTrigger(KeyInput.KEY\_LSHIFT);

 public final static String MAPPING\_COLOR\_OF\_FIRST = "Change first body's color";

 public final static String MAPPING\_COLOR\_OF\_SECOND = "Change second body's color";

 public final static String MAPPING\_COLOR\_OF\_THIRD = "Change third body's color";

 public final static String MAPPING\_ROTATE\_OF\_FIRST = "Change first body's position";

 public final static String MAPPING\_ROTATE\_OF\_SECOND = "Change second body's position";

 public final static String MAPPING\_ROTATE\_OF\_THIRD = "Change third body's position";

 public final static String MAPPING\_BEGIN = "Reset to default";

 public final static String MAPPING\_REMOVING = "Removing hats";

 public final static String MAPPING\_SHIFT = "Pressed LShift";

 public final static String MAPPING\_SLASH = "Pressed Slashkey";

 public final static String MAPPING\_COMMA = "Pressed comma";

 public final static String MAPPING\_POINT = "Pressed Points";

}

**ControlToRotate class**

/\*

 \* To change this license header, choose License Headers in Project Properties.

 \* To change this template file, choose Tools | Templates

 \* and open the template in the editor.

 \*/

package mygame;

import com.jme3.math.Vector3f;

import com.jme3.renderer.RenderManager;

import com.jme3.renderer.ViewPort;

import com.jme3.scene.Spatial;

import com.jme3.scene.control.AbstractControl;

/\*\*

 \*

 \* @author Pavel

 \*/

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) {

    }

}

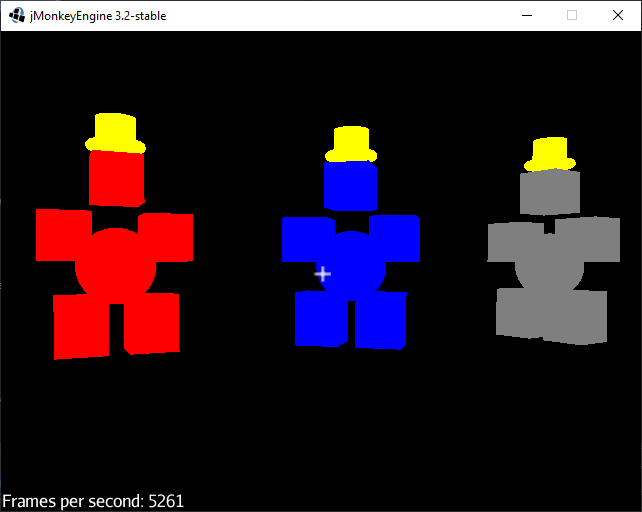


Рис. 1 Вигляд персонажів

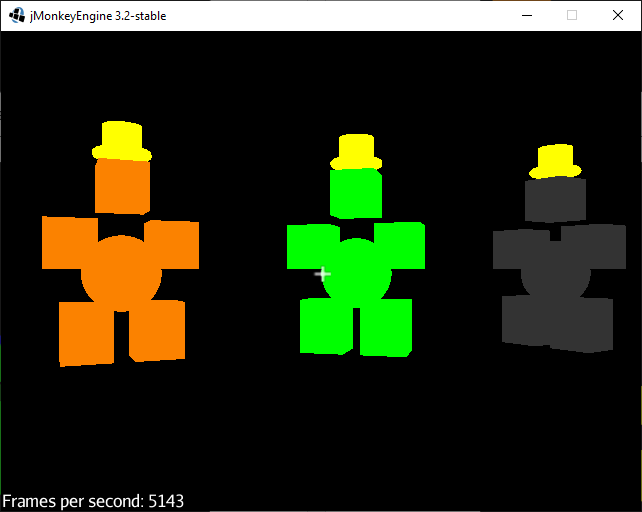


Рис. 2 Після зміни кольору

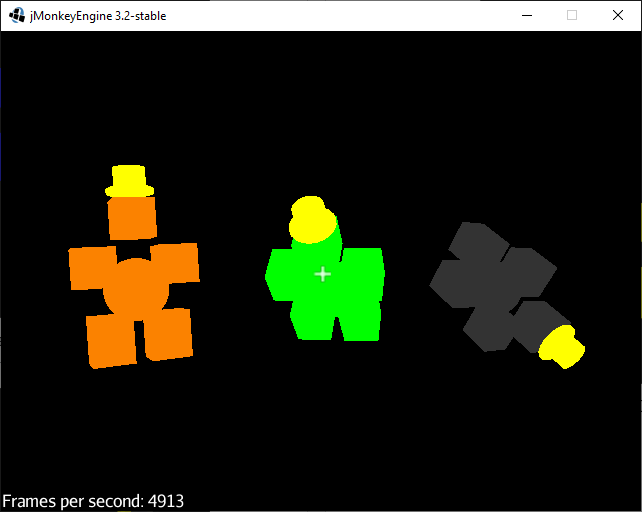


Рис. 3 Після обертання

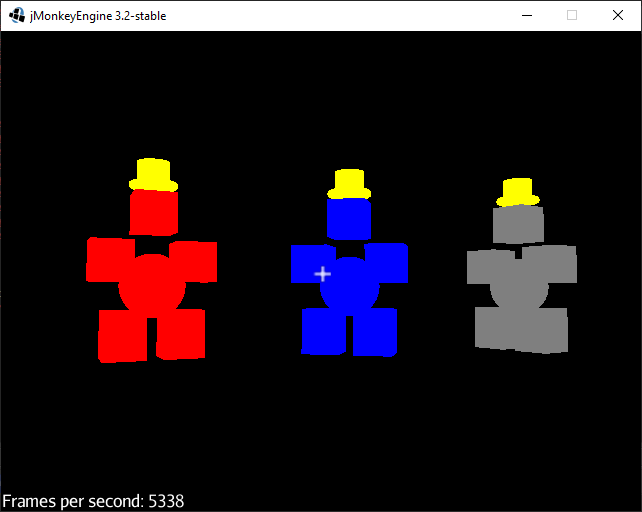


Рис. 4 Повернення до початкового стану

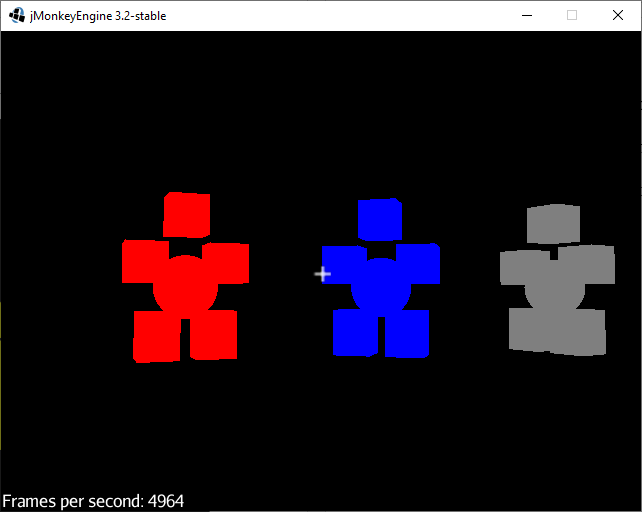


Рис. 5 Після видалення капелюхів

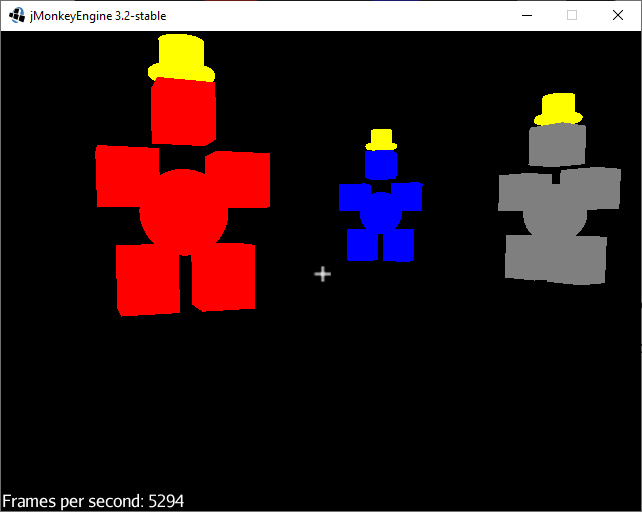


Рис. 6 Після змінення розмірів

Висновок: я дізнався про можливості взаємодії між користувачем і програмою у JME, за допомогою тригерів та застосував ці навички при створенні додатку. Також дізнався про можливість отримання даних про зреагований об’єкт та їх зміну.