KAIST

CS 482 Lab Session

Interactive Computer Graphics

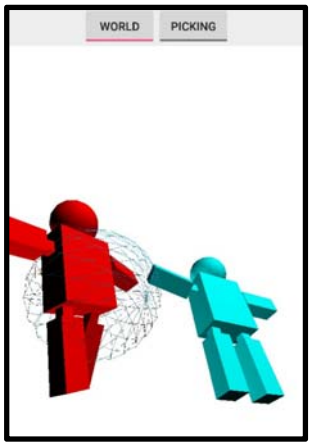
2016.11.2

VISUAL COMPUTING Lab

Goals

KAIST

- Color picking with Scene graph
  - Implement user manipulation code

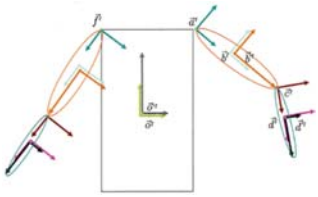


VISUAL COMPUTING Lab

Preliminaries

KAIST

- You should be understand
  - How to work the SceneGraph
  - How to make a color-coded scene
  - How to draw scene on the FrameBuffer in OpenGL
  - How to pick color from the FrameBuffer



3

VISUAL COMPUTING Lab

Pick OBJ with Color

KAIST

- Color mapping with ID
  - Make some color encoding method utilizing the ID of object: colorToId()
  - Store the color-ID pair
- Draw color-coded scene
  - Draw each object with coded color generated from object's ID
  - Color-coded scene is drawn on the FrameBuffer
- Pick object from the scene
  - Get the color on the position (x,y)
  - Find the object which is drawn with the picked color.

4

VISUAL COMPUTING Lab

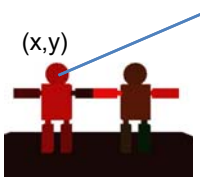
Pick OBJ with Color

KAIST

Draw
color-coded scene
on the FrameBuffer

RGB: (24,2,2)

(x,y)



Pick
Color of pixel at the
position (x,y)

Pick
Object from the
picked color

(24,2,2) ➔ id: 4 ➔ Head_Joint Node

IMPORTANT:
Color-coded scene never
expose to a user

Draw
The scene which is
show to user

5

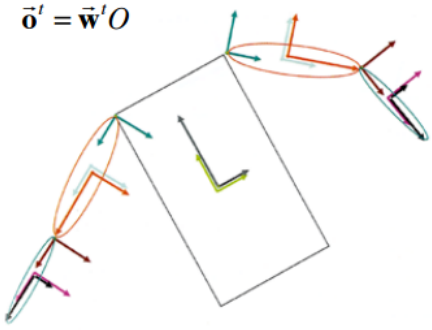
VISUAL COMPUTING Lab

Scene graph

KAIST

- Please remind the structure of SceneGraph

$\vec{\mathbf{o}}^t = \vec{\mathbf{w}}^t O$



$$\vec{\mathbf{o}}^t = \vec{\mathbf{w}}^t O$$
$$\vec{\mathbf{o}}'^t = \vec{\mathbf{o}}^t O'$$
$$\vec{\mathbf{a}}^t = \vec{\mathbf{o}}^t A$$
$$\vec{\mathbf{b}}^t = \vec{\mathbf{a}}^t B$$
$$\vec{\mathbf{b}}'^t = \vec{\mathbf{b}}^t B'$$
$$\vec{\mathbf{c}}^t = \vec{\mathbf{b}}^t C$$
$$\vec{\mathbf{d}}^t = \vec{\mathbf{c}}^t D$$
$$\vec{\mathbf{d}}'^t = \vec{\mathbf{d}}^t D'$$
$$\vec{\mathbf{f}}^t = \vec{\mathbf{o}}^t F$$

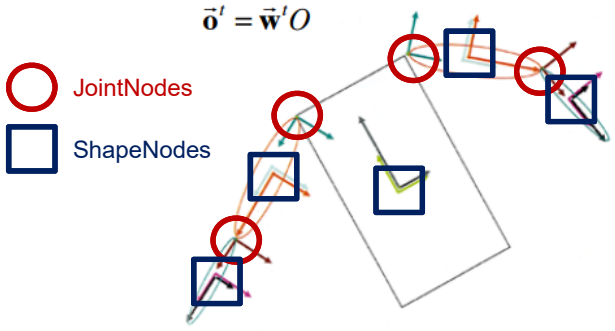
6

VISUAL COMPUTING Lab

Scene graph

KAIST

- Also, you should think about the different role of “JointNode” and “ShapeNode”



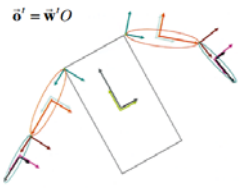
$$\vec{o}^t = \vec{w}^t O$$
$$\vec{o}^{tt} = \vec{o}^t O'$$
$$\vec{a}^t = \vec{o}^t A$$
$$\vec{b}^t = \vec{a}^t B$$
$$\vec{b}^{tt} = \vec{b}^t B'$$
$$\vec{c}^t = \vec{b}^t C$$
$$\vec{d}^t = \vec{c}^t D$$
$$\vec{d}^{tt} = \vec{d}^t D'$$
$$\vec{f}^t = \vec{o}^t F$$

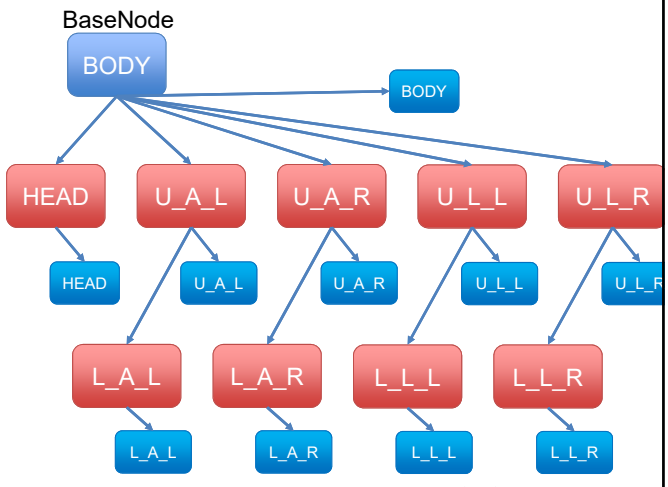
VISUAL COMPUTING Lab

Scene graph

KAIST

- Structure of MyRobot





Joint Node SgRbtNode

Shape Node SgGeometry ShapeNode

VISUAL COMPUTING Lab

Practice: result

KAIST

- Implement Color picker
  - Fill out the Picker class
  - Fill out the MyGLRenderer class
  - Replace the blue robot to the your own object (which is implemented at previous lab session)

- Motion events are already implemented
- World mode: is rendered with DEFAULT_SHADER
- Picking mode: is rendered with PICKING_SHADER (Color-coded scene)

VISUAL COMPUTING Lab

Practice: Task 1

KAIST

- Fill out the Picker class on the skeleton code

4

```
private void addToMap(int id, SgRbtNode node){
    //Task: add the pair of "id" and "node" into "idToRbtNodeMap".
}

private SgRbtNode find(int id){
    //Task: Find the JointNode utilizing the id from the idToRbtNodeMap
    return null;
}
```

5

idToColor & colorToId methods are already implemented, but you should understand how to work those methods

```
@Override
public boolean visit(SgShapeNode node){
    boolean result = true;
    idCounter++;
    for(int i = nodeStack.size() -1; i >= 0; --i){
        SgRbtNode asRbtNode = (SgRbtNode)nodeStack.get(i);
        if(asRbtNode != null){
            addToMap(idCounter, asRbtNode);
            break;
        }
    }
    float[] idColor = idToColor(idCounter);

    //Task: Draw the color-coded scene utilizing the generated idColor and drawer
    return result;
}
```

1

6

```
public SgRbtNode getRbtNodeAtXY(int x, int y){
    //Task: Read the color at position (x,y) from the FrameBuffer in openGL
    //After that, utilizing the color, find and return the JointNode which rendered at position (x,y)

    ByteBuffer query_color;
    int id = -1;
    return find(id);
}
```

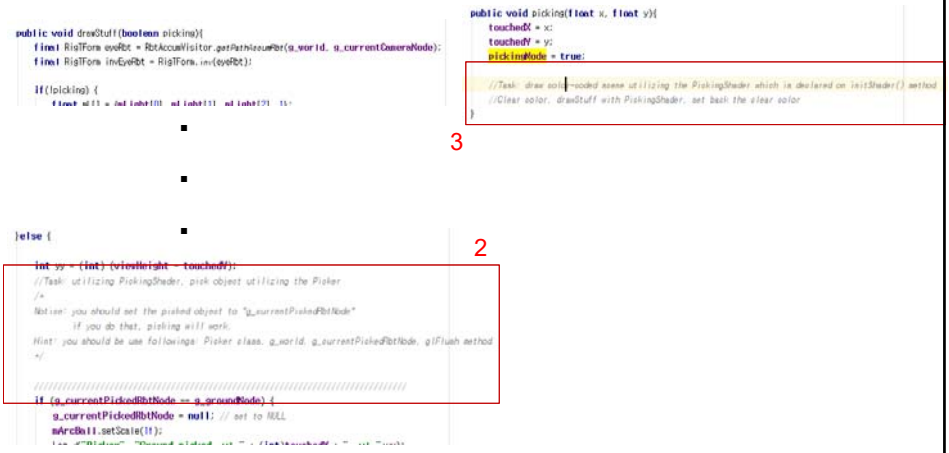
The number is represented recommended fill-out order

VISUAL COMPUTING Lab

Practice: Task 2

KAIST

- Fill out the MyGLRenderer class on the skeleton code



VISUAL COMPUTING Lab

Practice: Task3

KAIST

- MyGLRenderer: Replace the “g_robotNode2” to the your own object which is implemented at previous lab session

If your object do not work as a MyRobot object, you should modify the center position of nodes on your own object.

Also you should refer the pages about SceneGraph structure.

VISUAL COMPUTING Lab

Practice: Ref

KAIST

- Please check the MyGLSurfaceView
  - queueEvent
    - queueEvent is provided class in openGL ES
    - It is run on the rendering thread.
    - it is need for Event handling with your renderer

```
//queueEvent is run on the rendering thread,
//it is need for communicate with your renderer
queueEvent(new Runnable() {
    @Override
    public void run() {
        mRenderer.picking(mPreviousX,mPreviousY);
    }
});
```


VISUAL COMPUTING Lab

Advance

KAIST

- Let's trying to use embedded sensors on Android phone!

(for participators who already implement three tasks at the previous lab session)

- We can use three kind of IMU (Inertia Measurement Unit) measurements from an android phone.

- Linear Accelerometer
- Angular Accelerometer (Gyroscope)
- Orientation (roll, pitch, yaw)

VISUAL COMPUTING Lab

Advance

KAIST

- Let's trying to use embedded sensors on Android phone!

Orientation code is already implemented

VISUAL COMPUTING Lab

Advance

KAIST

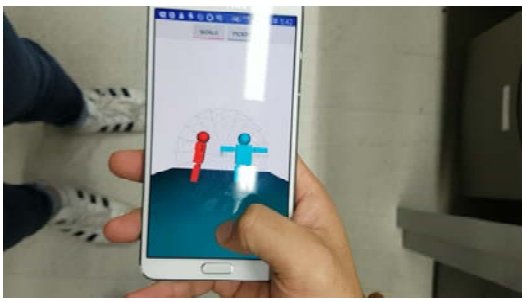
- Let's trying to use embedded sensors on Android phone!

VISUAL COMPUTING Lab

Advance

KAIST

- Let's trying to use embedded sensors on Android phone!
  - Utilizing the measured Orientation values (roll, pitch, yaw of your android phone),
  - Rotate red robot when you touched a finger on screen



17

VISUAL COMPUTING Lab