Laboratory

Guide

Schedule

SciVis

Date	Week	Session
20/02	2	1 - OpenGL
27/02	3	2.1 - Voxel
06/03	4	2.1 - Voxel
13/03	5	2.2 - IVR
20/03	6	2.3 - DVR

InfoVis

Date	Núm.	Session
03/04	8	3.1 Presentation Design
24/04	9	3.2 Best Practices
08/05	10	4.1 Special Vis
15/05	11	4.2 Special Vis
22/05	12	4.3 Interaction

Lab 1

OpenGL

IDE

- You can use any IDE or editor (ex. Sublime)

- But, the recommendation is :

  - **IntelliJ IDEA** :
    - https://www.jetbrains.com/idea/download/

Groovy

Language

Overview

- It is an agile and **dynamic language** for the Java Virtual Machine
- Builds upon the **strengths of Java** but has **additional power features** inspired by languages like Python, Ruby and Smalltalk
- Makes **modern programming features** available to Java developers with **almost-zero learning curve**

Benefits

- Provides the ability to **statically type check** and **statically compile** your code for robustness and performance

- Supports **Domain-Specific Languages** and other compact syntax so your code becomes **easy to read and maintain**

- Makes writing shell and build scripts easy with its **powerful processing primitives** and OO abilities

Why Groovy?

- Increases developer productivity by **reducing scaffolding code** when developing web, GUI, database or console applications

- **Simplifies testing** by supporting unit testing and mocking out-of-the-box

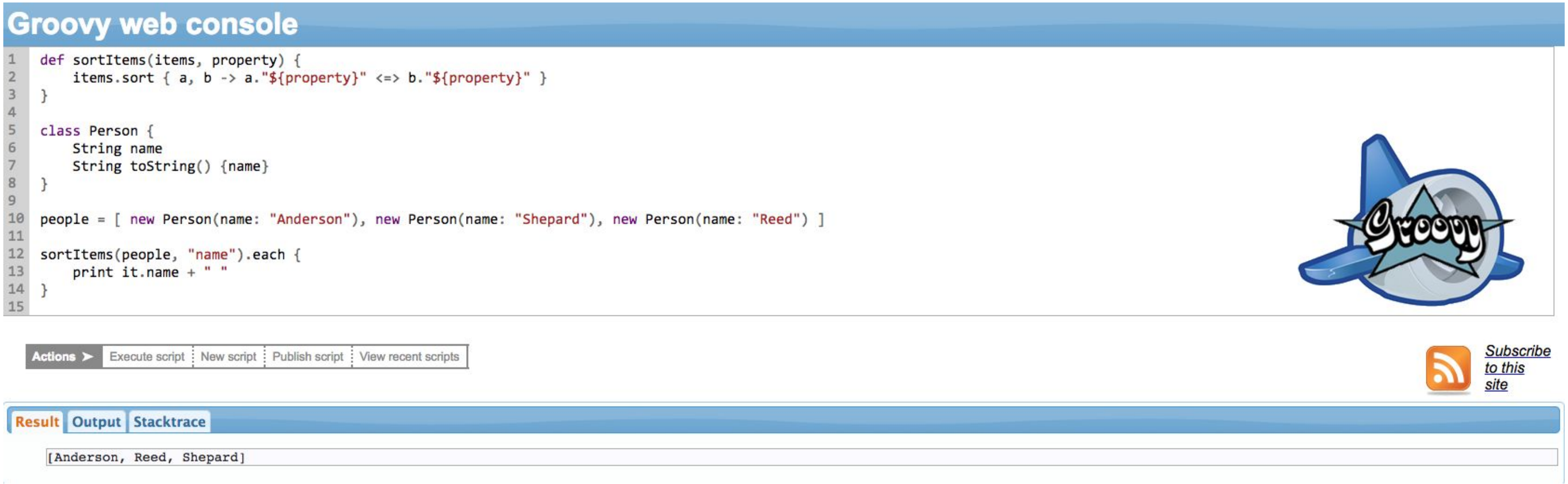
- Seamlessly **integrates with all existing Java classes and libraries**

Script Example

```
def sortItems(items, property) {
    items.sort { a, b ->
             a."${property}" <=> b."${property}"
    }
}

class Person {
    String name
    String toString() {name}
}

people = [ new Person(name: "Anderson"),
                 new Person(name: "Shepard"),
                 new Person(name: "Reed") ]

sortItems(people, "name").each {
    print it.name + " "
}

//Result : [Anderson, Reed, Shepard]
//Output : "Anderson Reed Shepard "
```

Test it!

Browser : https://groovyconsole.appspot.com/



```
def sortItems(items, property) {
    items.sort { a, b -> a."${property}" <=> b."${property}" }
}

class Person {
    String name
    String toString() {name}
}

people = [ new Person(name: "Anderson"), new Person(name: "Shepard"), new Person(name: "Reed") ]

sortItems(people, "name").each {
    print it.name + " "
}

```

```
[Anderson, Reed, Shepard]
```

Explanation

- The declaration of a method to sort a collection by a property of the objects

- The declaration of a class with a single property.
  - Getters and setters are constructed automatically

- The creation of a list, all on one line, without having to create a collection, etc

- The list is ordered according to the name of the specified property.

Gradle

- A general purpose **build system**
- Comes with a rich build DSL (Groovy)
- Supports *build-by-convention* principle
- Very flexible and extensible
- Built-in plugins for Java, Groovy, etc.

Links

- https://docs.gradle.org/current/userguide/tutorial_groovy_projects.html
- https://www.infoq.com/articles/new-groovy-20
- https://docs.gradle.org/current/userguide/application_plugin.html
- http://es.slideshare.net/buzdin/gradle-introduction-9633872

JOGL

Library

Instructions

1. Take a look on **lab1.zip**

2. Import it on **IntelliJ IDEA**

3. Follow the **OneTriangle** example

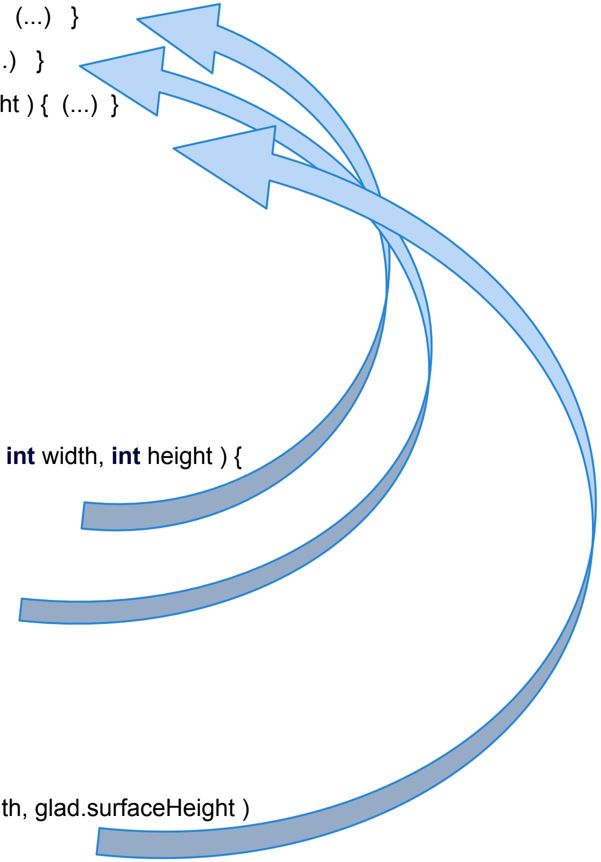
4. Follow the **Gears** example

Project

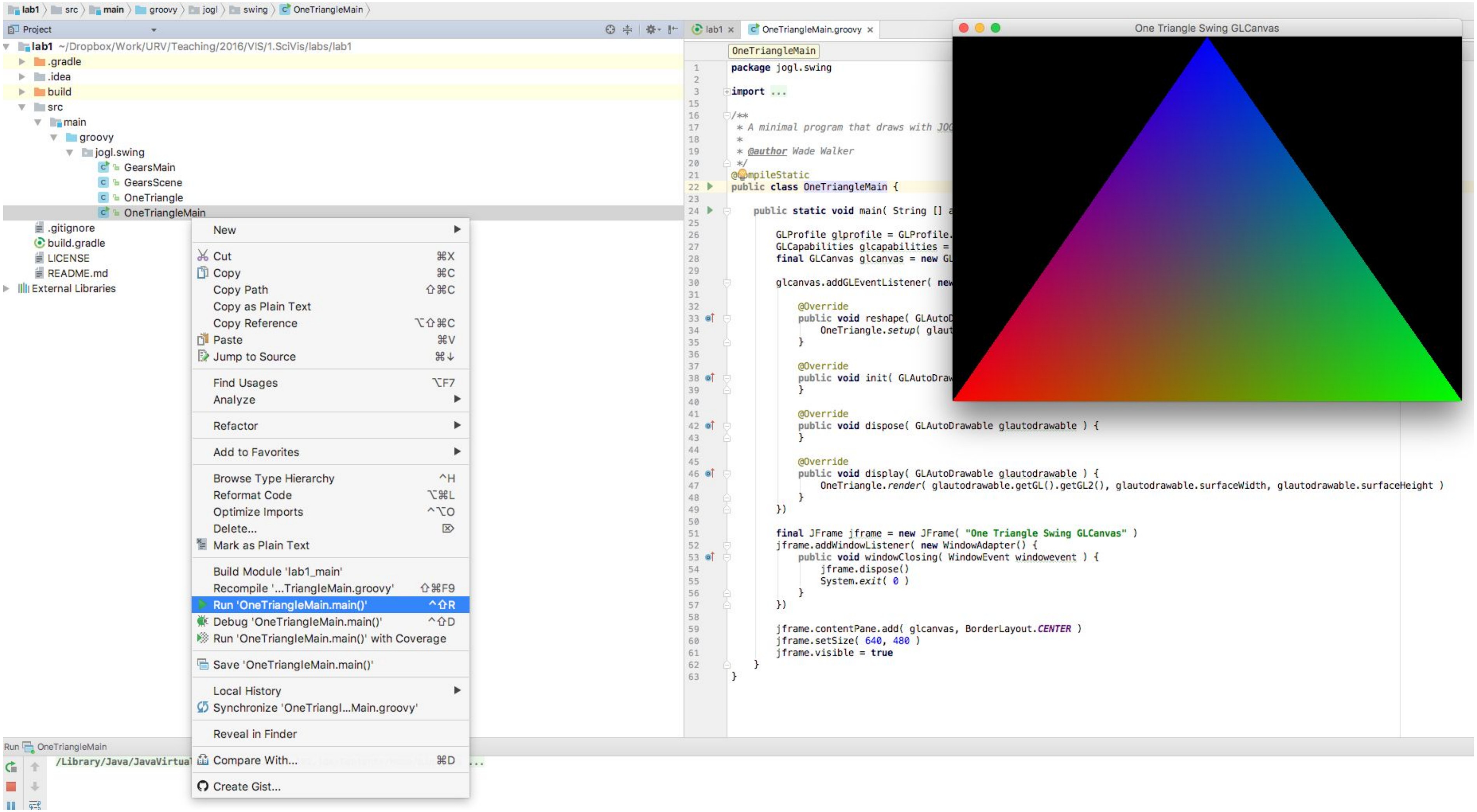
- Sample Groovy/Gradle project with JOGL
- One Triangle Example
- Gears Classic Example

Structure

```
public class ExampleScene {
   protected static void setup( GL2 gl2, int width, int height ) {   (...)   }
  protected static void init( GL2 gl2, int width, int height ) {   (...)   }
   protected static void render( GL2 gl2, float width, float height ) {  (...)  }
}



public class ExampleMain {
   public static void main( String [] args ) {
       (...)
       glcanvas.addGLEventListener( new GLEventListener() {
           public void reshape( GLAutoDrawable glad, int x, int y, int width, int height ) {
               scene.setup( glad.getGL().getGL2(), width, height )
           }
           public void init( GLAutoDrawable glad ) {
              scene.init( glad.getGL().getGL2())
          }
           public void dispose( GLAutoDrawable glad ) { }
           public void display( GLAutoDrawable glad ) {
               scene.render( glad.getGL().getGL2(), glad.surfaceWidth, glad.surfaceHeight )
           }
       })
   }
}
```



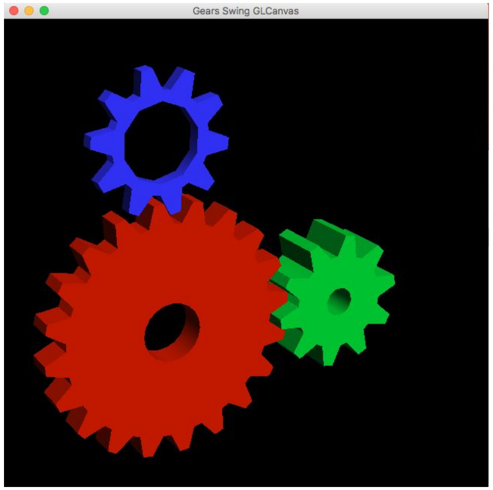
Instructions



```
package jogl.swing

import ...

/**
 * A minimal program that draws with JOG
 *
 * @author Wade Walker
 */
@ompileStatic
public class OneTriangleMain {

    public static void main( String [] a

        GLProfile glprofile = GLProfile.
        GLCapabilities glcapabilities =
        final GLCanvas glcanvas = new GL

        glcanvas.addGLEventListener( new

            @Override
            public void reshape( GLAutoD
                OneTriangle.setup( glaut
            }

            @Override
            public void init( GLAutoDraw
            }

            @Override
            public void dispose( GLAutoDrawable glautodrawable ) {
            }

            @Override
            public void display( GLAutoDrawable glautodrawable ) {
                OneTriangle.render( glautodrawable.getGL().getGL2(), glautodrawable.surfaceWidth, glautodrawable.surfaceHeight )
            }
        })

        final JFrame jframe = new JFrame( "One Triangle Swing GLCanvas" )
        jframe.addWindowListener( new WindowAdapter() {
            public void windowClosing( WindowEvent windowevent ) {
                jframe.dispose()
                System.exit( 0 )
            }
        })

        jframe.contentPane.add( glcanvas, BorderLayout.CENTER )
        jframe.setSize( 640, 480 )
        jframe.visible = true
    }
}
```

Instructions

- Import the gradle project or via CLI :
    - **Compile** : *gradle compileGroovy*
    - **Execute** : *gradle run*



Links

- https://jogamp.org/wiki/index.php/Jogl_Tutorial#JogAmp.27s_Static
- http://jogamp.org/wiki/index.php/Release_2.3.1
- https://jogamp.org/wiki/index.php/Using_JOGL_in_AWT_SWT_and_Swing

Lab 2

Voxels

Instructions

1. Take a look on **lab2.zip**

2. Import it on **IntelliJ IDEA**

3. Fill the gaps in the **PhantomModel** class

4. **Execute** the gradle project

Exercise 1

- Title : Phantom Model
- First Lab Exercise
- Deadline : Next Session

Explore

- **Scene** class : JOGL code ready to render and interact with the volume
- **Model** class : generic abstract volume class
- **PhantomModel** class : unfinished implementation to generate the Phantom model by using the heat points information

Fill the Gaps

- **Distance** method : calculate the distance between two tridimensional points
- **Contribution** method : calculate the value of each voxel in the model based on the heat points and the distance between them
- **CreateVoxelModel** method : generate the volume by iterating and calculating the contribution of each model

Test it!

- Main class voxel.**Test**

- Gradle **run** (includes compile)

- **Key up** to look all the slices (256)

Delivery

- Once you are able to observe the heat points

- Zip your project

- Upload to the proper model task

Lab 3

Voxels (II)

Instructions

1. Take a look on **lab3.zip**

2. Import it on **IntelliJ IDEA**

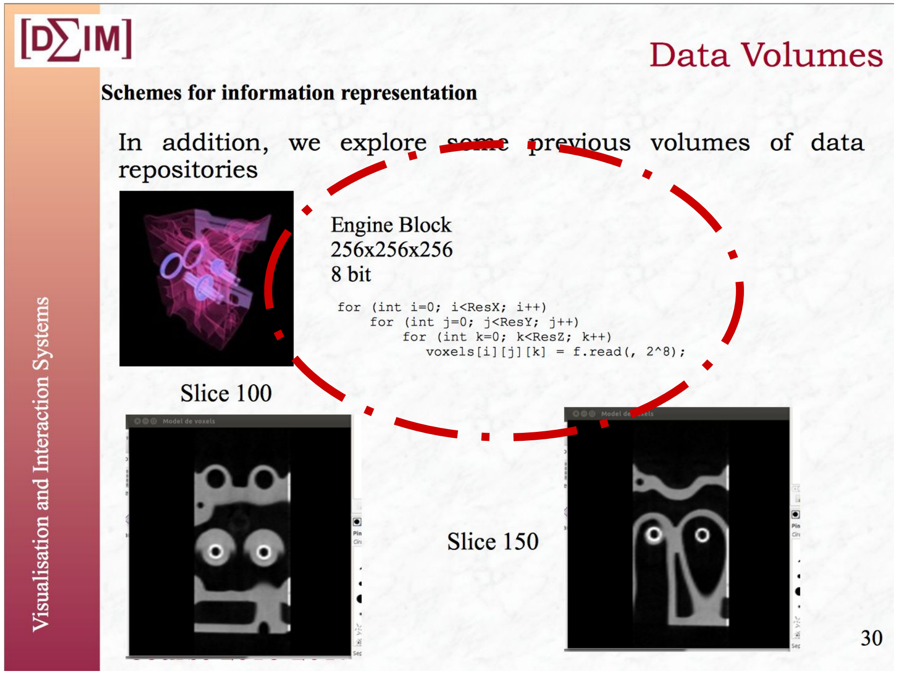
3. Fill the gaps in the **Model** class

4. **Execute** the gradle project

Models

- Review the theory (2.1, pages 30-33)

- Remember each model has :
  - Size : different sizes
  - Value : 8 or 16 bits



[DΣIM]

Data Volumes

Schemes for information representation

In addition, we explore some previous volumes of data repositories

Engine Block
256x256x256
8 bit

```
for (int i=0; i<ResX; i++)
     for (int j=0; j<ResY; j++)
          for (int k=0; k<ResZ; k++)
             voxels[i][j][k] = f.read(, 2^8);
```

Slice 100

Slice 150

Visualisation and Interaction Systems

30

Models

- Visualize each model :
  - nucleon.raw
  - marschnerlobb.raw
  - fuel.raw
  - Engine.raw
  - tomato.raw
  - present.dat

- Finally, post your renders in the forum!

Lab 4

Voxels (III)

Instructions

1. Take a look on **lab4.zip**

2. Import it on **IntelliJ IDEA**

3. **Execute** the gradle project

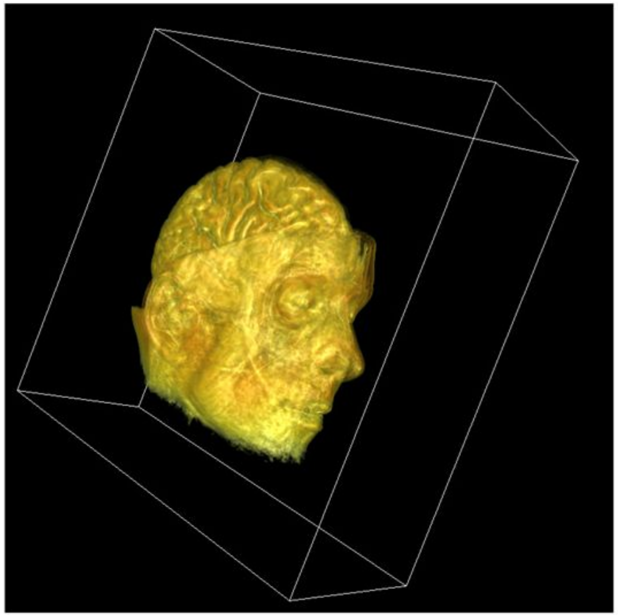
4. **Extend** the project to extract information

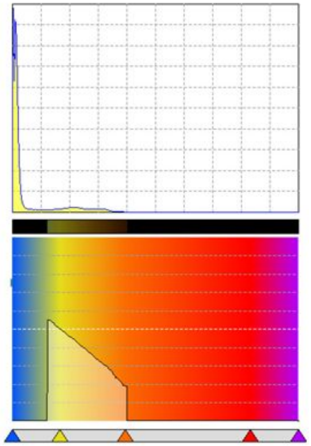
Context

- The subject of this virtual session is to measure how affect compression to data structures
- To simplify the problem we will just make the study a slice image level from the volume of data
- Work with the selected models: tomato, present, and engine

Histogram

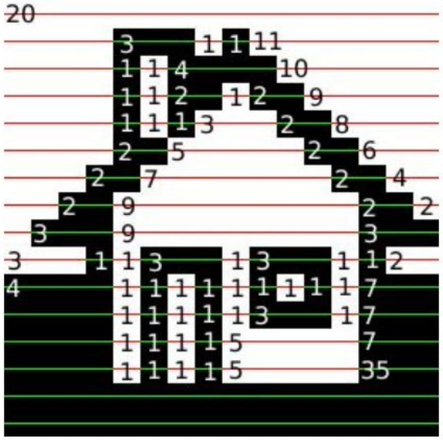
- By using the lab4 implementation, extend this OpenGL code in order to recollect the outstanding information
- Show the information in a proper visualisation
- Example :





Runlength

- By using the lab4 implementation, extend this OpenGL code in order to recollect the outstanding information
- Show a proper visualisation of a selected slice adding the compression information
- Example :



Runlength

- Lastly, calculate for each model which is
  - the % of void voxels before and after the compression
  - the resultant gain in each case

Lab 5

Voxels (Last)

Instructions

1. Recover the models from previous labs

2. For example from **lab4.zip**

3. In the folder **src/main/resources/models**

4. **Download** a tool : ParaView or VolView

Exercise 2

- Title : Volume Tools

- Second Lab Exercise

- Deadline : End of Course

Objectives

- Learn about “already built” tools that allows us to visualise data volumes

- Apply the operations that have been explained in the theory sessions

Decisions

- This will allow us to explore a data volume
- In particular, your first decision is to make a choice between one of the following volumes of data we have worked with :
    - Engine, Tomato, or Present.

Decisions

- The second decision is to make a choice between two recommended tools to work with

- In particular you can use: **ParaView** or **Volview**

- Then, you need to install the tool and review a tutorial to understand how to use the environment and explore data volumes

Procedure

1. Load the data volume in the chosen tool/environment

2. Implement the necessary strategies to interpret the volume information

3. Discuss the results of each strategy with emphasis on the parameters used

Deliverable

- Finally, make a report to deliver your work and to show the results that have been reached in each case

- The results obtained will be basically pictures and comments of each step in your process

TO BE CONTINUED