

TE : Ajp - Another Java Plotter

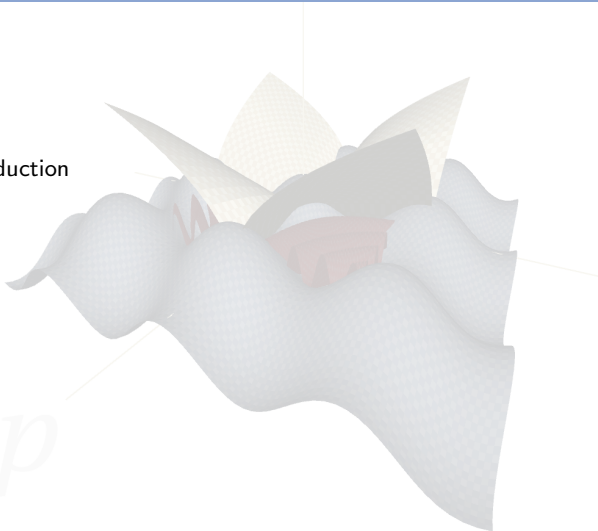
Lozi Jean-Pierre

4 juin 2008

Ajp

Sommaire

- Introduction



Sommaire

- Introduction
- Interface et Utilisation

Ajp

Sommaire

- Introduction
- Interface et Utilisation
- Implémentation

Ajp

Sommaire

- Introduction
- Interface et Utilisation
- Implémentation
- Vue rapide de l'API

Ajp

Sommaire

- Introduction
- Interface et Utilisation
- Implémentation
- Vue rapide de l'API
- Exemples d'utilisation

Ajp

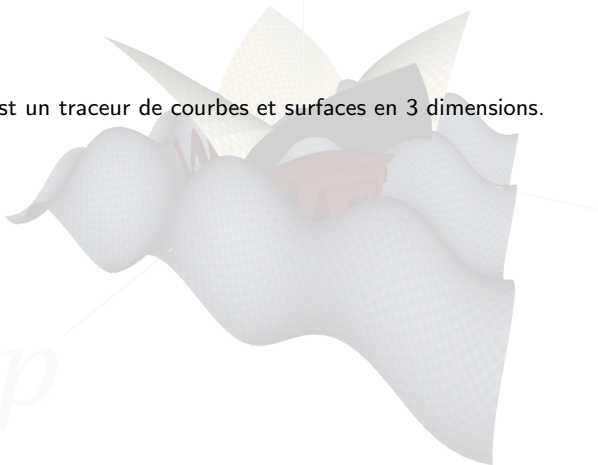
Sommaire

- Introduction
- Interface et Utilisation
- Implémentation
- Vue rapide de l'API
- Exemples d'utilisation
- Conclusion

Ajp

Présentation

- Ajp est un traceur de courbes et surfaces en 3 dimensions.



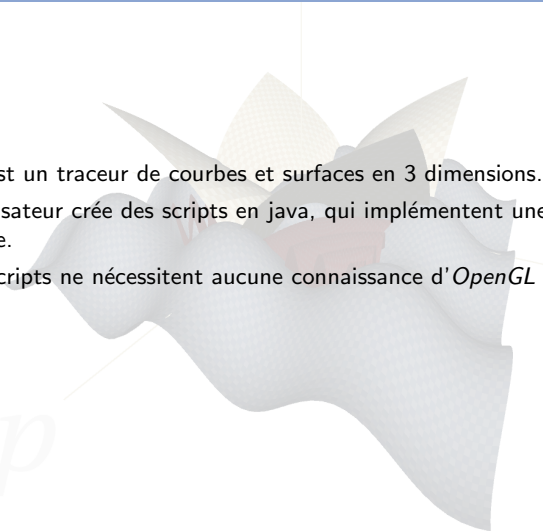
Ajp

Présentation

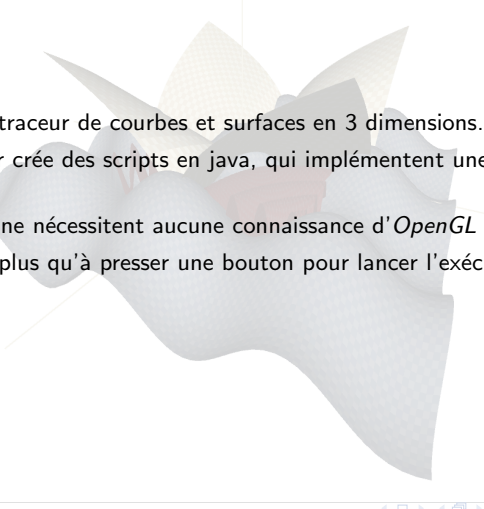
- Ajp est un traceur de courbes et surfaces en 3 dimensions.
- L'utilisateur crée des scripts en java, qui implémentent une interface simple.

Ajp

Présentation

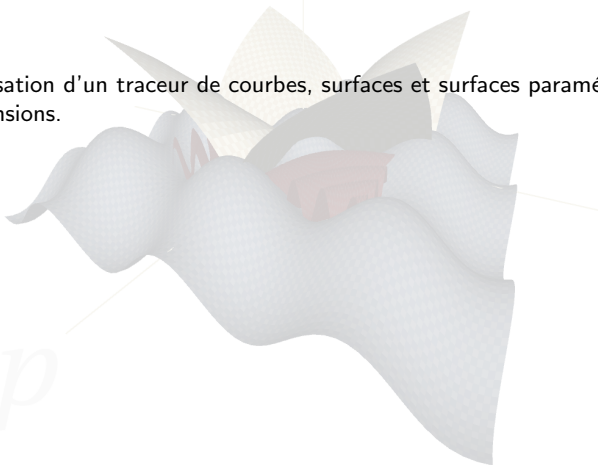
- 
- Ajp est un traceur de courbes et surfaces en 3 dimensions.
 - L'utilisateur crée des scripts en java, qui implémentent une interface simple.
 - Ces scripts ne nécessitent aucune connaissance d'*OpenGL* (ni de *Swing*).

Présentation

- 
- Ajp est un traceur de courbes et surfaces en 3 dimensions.
 - L'utilisateur crée des scripts en java, qui implémentent une interface simple.
 - Ces scripts ne nécessitent aucune connaissance d'*OpenGL* (ni de *Swing*).
 - Il n'a alors plus qu'à presser une bouton pour lancer l'exécution de son script.

Objectifs du projet

- Réalisation d'un traceur de courbes, surfaces et surfaces paramétrées en 3 dimensions.



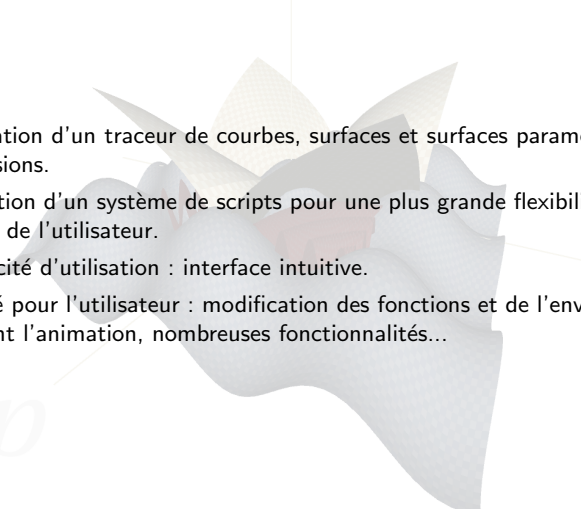
Objectifs du projet

- Réalisation d'un traceur de courbes, surfaces et surfaces paramétrées en 3 dimensions.
- Utilisation d'un système de scripts pour une plus grande flexibilité du point de vue de l'utilisateur.

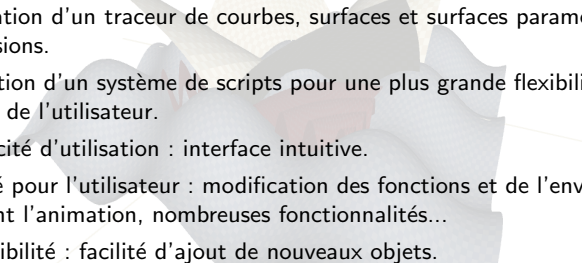
Objectifs du projet

- Réalisation d'un traceur de courbes, surfaces et surfaces paramétrées en 3 dimensions.
- Utilisation d'un système de scripts pour une plus grande flexibilité du point de vue de l'utilisateur.
- Simplicité d'utilisation : interface intuitive.

Objectifs du projet

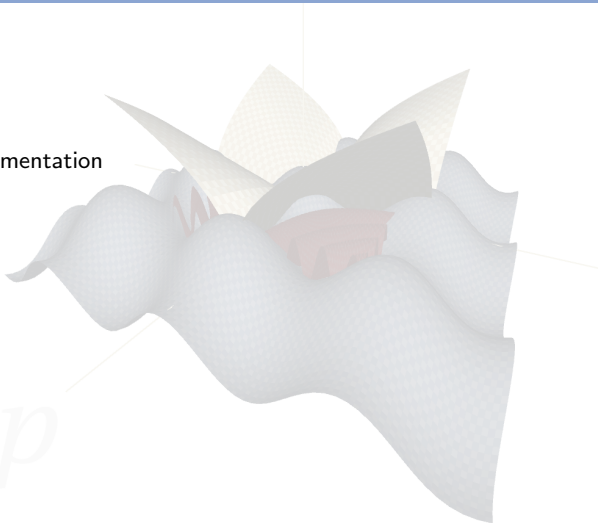
- 
- Réalisation d'un traceur de courbes, surfaces et surfaces paramétrées en 3 dimensions.
 - Utilisation d'un système de scripts pour une plus grande flexibilité du point de vue de l'utilisateur.
 - Simplicité d'utilisation : interface intuitive.
 - Liberté pour l'utilisateur : modification des fonctions et de l'environnement pendant l'animation, nombreuses fonctionnalités...

Objectifs du projet

- 
- Réalisation d'un traceur de courbes, surfaces et surfaces paramétrées en 3 dimensions.
 - Utilisation d'un système de scripts pour une plus grande flexibilité du point de vue de l'utilisateur.
 - Simplicité d'utilisation : interface intuitive.
 - Liberté pour l'utilisateur : modification des fonctions et de l'environnement pendant l'animation, nombreuses fonctionnalités...
 - Extensibilité : facilité d'ajout de nouveaux objets.

Contraintes (1)

■ Implémentation



Contraintes (1)

■ Implémentation

- de la compilation des classes pendant l'exécution.
- du chargement des classes pendant l'exécution.

Ajp

Contraintes (1)

- Implémentation
 - de la compilation des classes pendant l'exécution.
 - du chargement des classes pendant l'exécution.
- Ne mettre que certaines classes à la disposition de l'utilisateur :

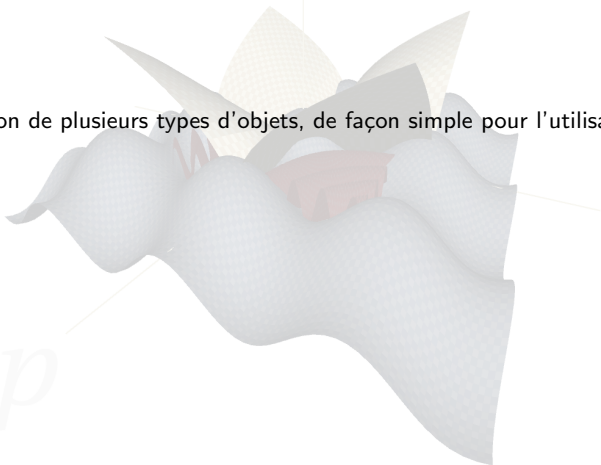
Ajp

Contraintes (1)

- Implémentation
 - de la compilation des classes pendant l'exécution.
 - du chargement des classes pendant l'exécution.
- Ne mettre que certaines classes à la disposition de l'utilisateur :
 - L'utilisateur ne doit pas avoir besoin de connaître le fonctionnement interne du logiciel.
 - L'utilisateur ne doit pas avoir besoin de connaître les *APIs OpenGL* ou *Swing*.

Contraintes (2)

- Gestion de plusieurs types d'objets, de façon simple pour l'utilisateur :



Contraintes (2)

- Gestion de plusieurs types d'objets, de façon simple pour l'utilisateur :
 - Courbes et surfaces paramétrées, surfaces “classiques” données par des équations explicites.
 - L'utilisateur ne doit avoir qu'à spécifier :

Contraintes (2)

- Gestion de plusieurs types d'objets, de façon simple pour l'utilisateur :
 - Courbes et surfaces paramétrées, surfaces “classiques” données par des équations explicites.
 - L'utilisateur ne doit avoir qu'à spécifier :
 - L'équation de la courbe.
 - L'intervalle des paramètres, ou bornes du tracé.
 - Les paramètres “esthétiques” : couleur de fond, du tracé, épaisseur du trait...

Quelques fonctions implémentées

- Possibilité de charger des fichiers java ou des classes précompilées.

Ajp

Quelques fonctions implémentées

- Possibilité de charger des fichiers java ou des classes précompilées.
- Possibilité de spécifier plusieurs graphiques ou fenêtres dans une seule classe.

Ajp

Quelques fonctions implémentées

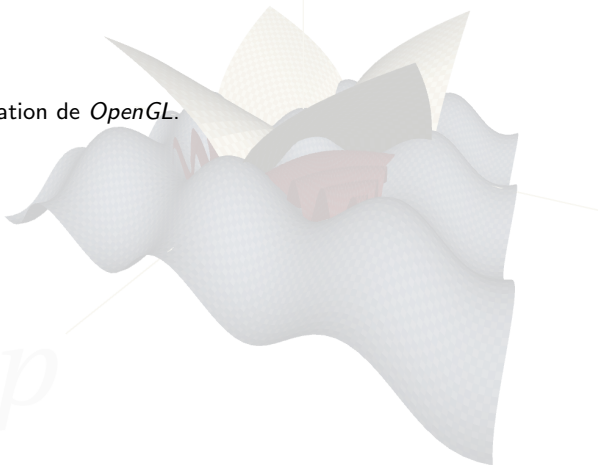
- Possibilité de charger des fichiers java ou des classes précompilées.
- Possibilité de spécifier plusieurs graphiques ou fenêtres dans une seule classe.
- Possibilité de rotation du graphique, zoom avant, arrière...

Quelques fonctions implémentées

- Possibilité de charger des fichiers java ou des classes précompilées.
- Possibilité de spécifier plusieurs graphiques ou fenêtres dans une seule classe.
- Possibilité de rotation du graphique, zoom avant, arrière...
- Selection de plusieurs couleurs différentes pour les différents tracés.

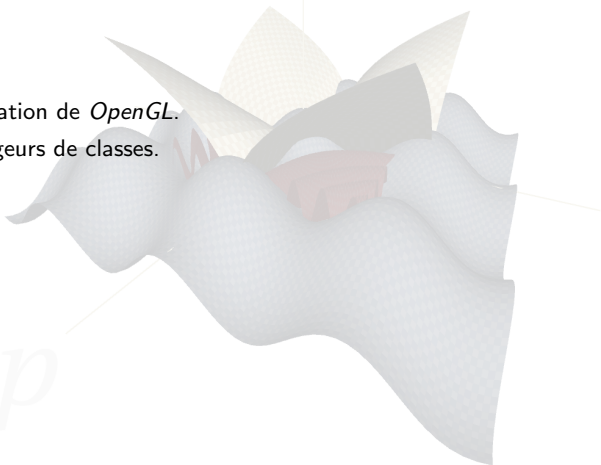
Code

- Utilisation de *OpenGL*.



Code

- Utilisation de *OpenGL*.
- Chargeurs de classes.



Code

- Utilisation de *OpenGL*.
- Chargeurs de classes.
- Utilisation de threads.

Ajp

Code

- Utilisation de *OpenGL*.
- Chargeurs de classes.
- Utilisation de threads.
- Code fonctionnel, pouvant être amélioré :

Ajp

Code

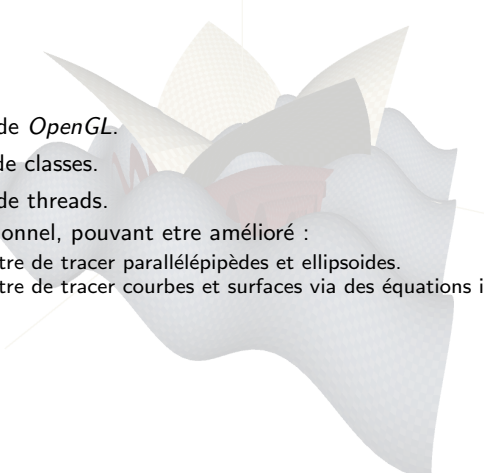
- 
- Utilisation de *OpenGL*.
 - Chargeurs de classes.
 - Utilisation de threads.
 - Code fonctionnel, pouvant être amélioré :
 - permettre de tracer parallélépipèdes et ellipsoïdes.
 - permettre de tracer courbes et surfaces via des équations implicites.

Diagramme de classes

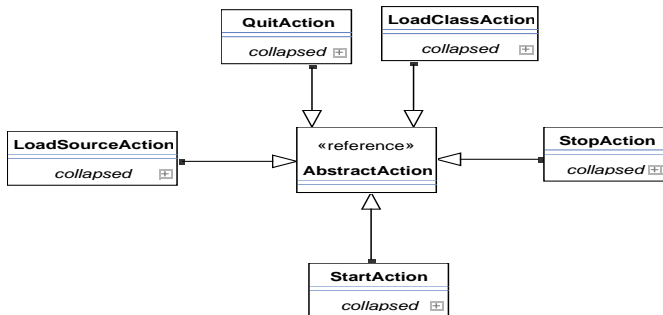


Diagramme de classes

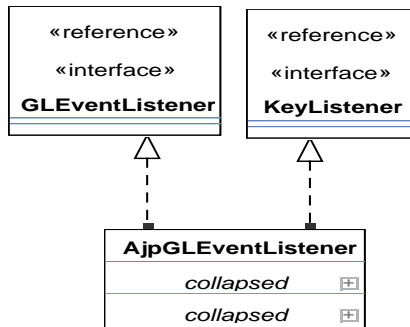


Diagramme de classes

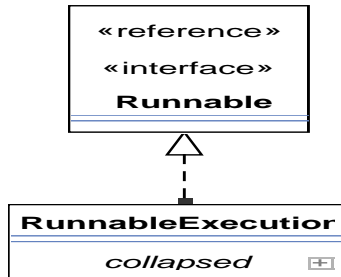


Diagramme de classes

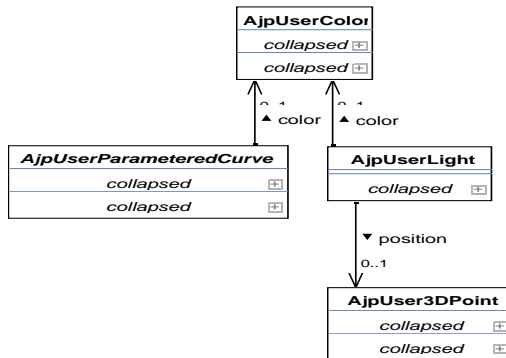
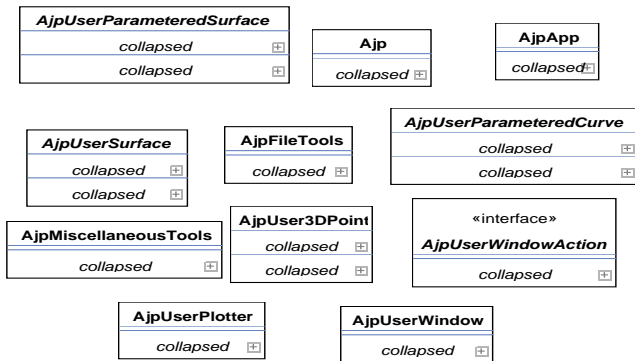


Diagramme de classes



Interface et Utilisation

- L'interface est intuitive. Elle propose deux fonctions principales :

Ajp

Interface et Utilisation

- L'interface est intuitive. Elle propose deux fonctions principales :
 - Le chargement d'un script

Ajp

Interface et Utilisation

- L'interface est intuitive. Elle propose deux fonctions principales :
 - Le chargement d'un script
 - sous forme d'un fichier java.
 - sous forme d'une classe compilée.

Interface et Utilisation

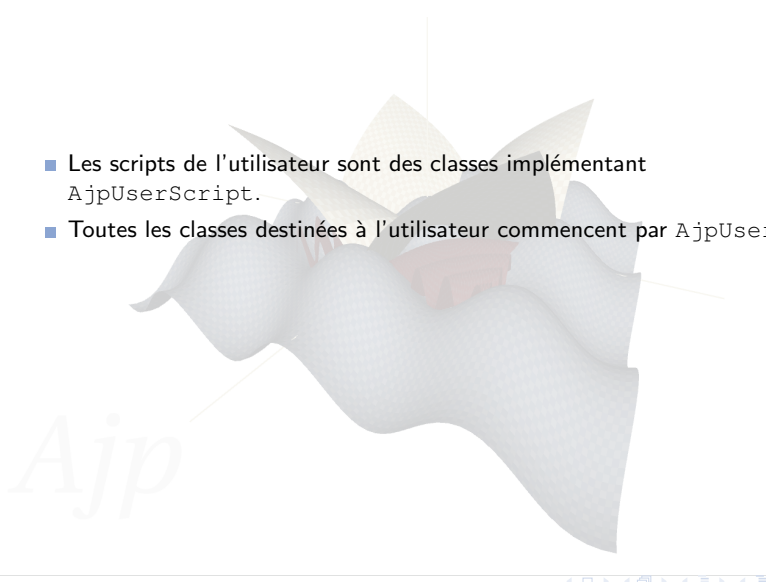
- L'interface est intuitive. Elle propose deux fonctions principales :
 - Le chargement d'un script
 - sous forme d'un fichier java.
 - sous forme d'une classe compilée.
 - Le lancement de l'exécution du script.

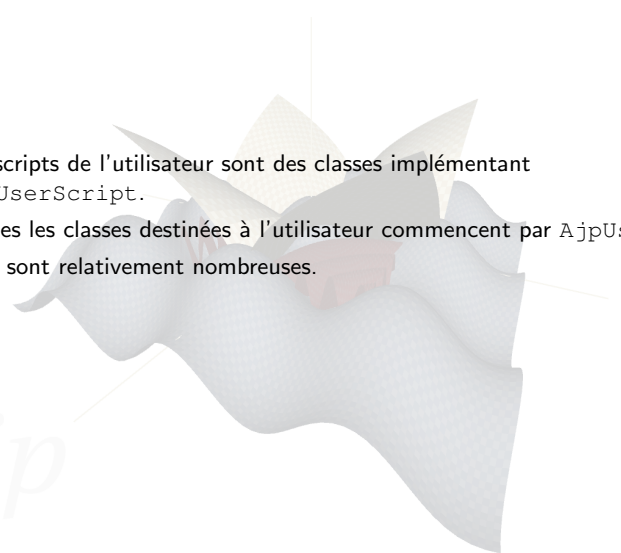
Interface et Utilisation



- Les scripts de l'utilisateur sont des classes implémentant `AjpUserScript`.

Ajp

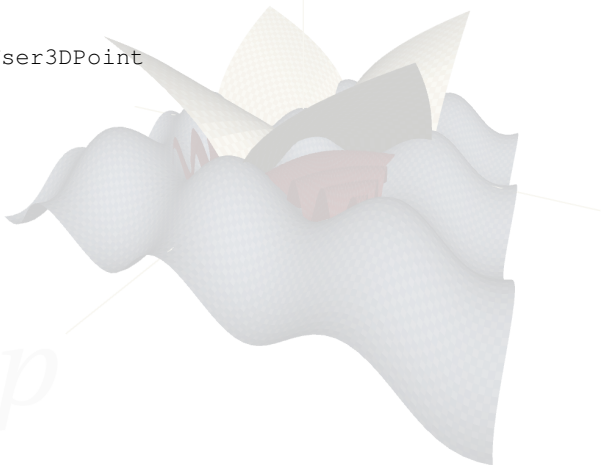
- 
- Les scripts de l'utilisateur sont des classes implémentant `AjpUserScript`.
 - Toutes les classes destinées à l'utilisateur commencent par `AjpUser`.

- 
- Les scripts de l'utilisateur sont des classes implémentant `AjpUserScript`.
 - Toutes les classes destinées à l'utilisateur commencent par `AjpUser`.
 - Elles sont relativement nombreuses.

- Les scripts de l'utilisateur sont des classes implémentant `AjpUserScript`.
- Toutes les classes destinées à l'utilisateur commencent par `AjpUser`.
- Elles sont relativement nombreuses.
- On ne présentera pas toutes leurs fonctions ici.

API - Classes

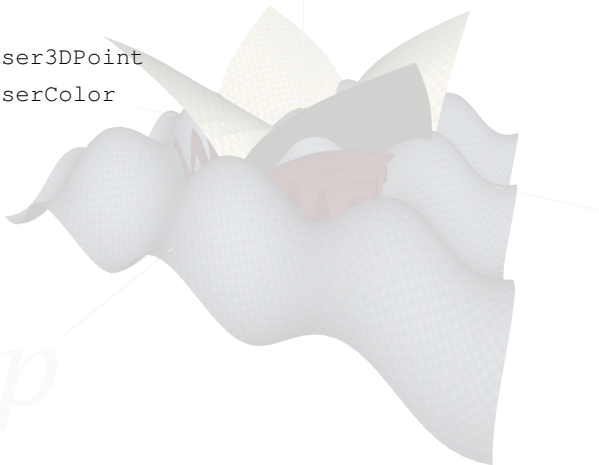
1 AjpUser3DPoint



API - Classes

1 AjpUser3DPoint

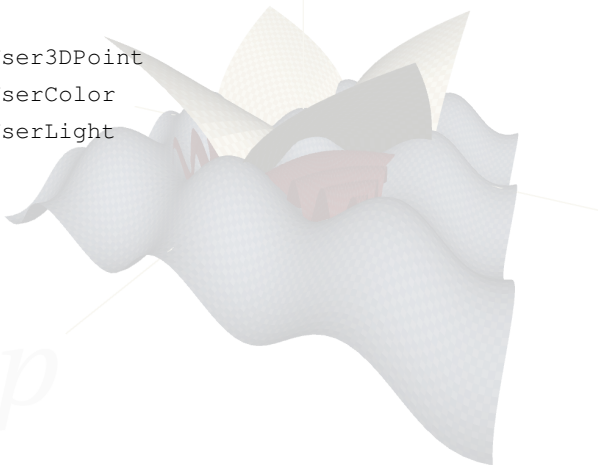
2 AjpUserColor



Ajp

API - Classes

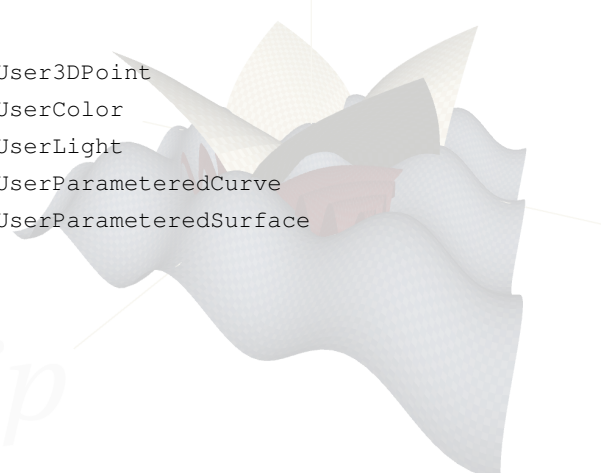
- 1 AjpUser3DPoint
- 2 AjpUserColor
- 3 AjpUserLight



API - Classes

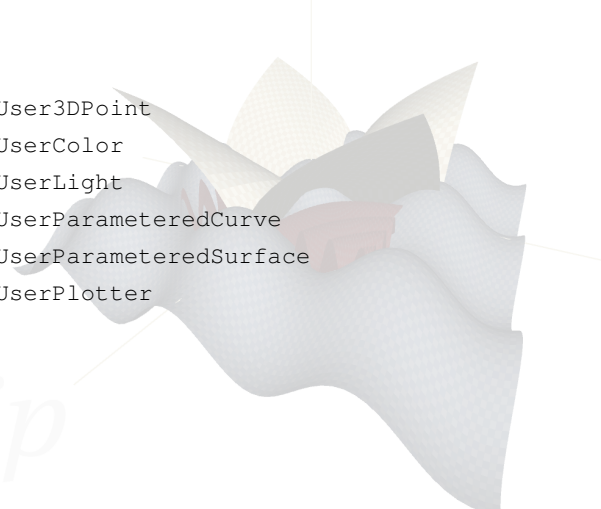
- 
- 1 AjpUser3DPoint
 - 2 AjpUserColor
 - 3 AjpUserLight
 - 4 AjpUserParameteredCurve

API - Classes

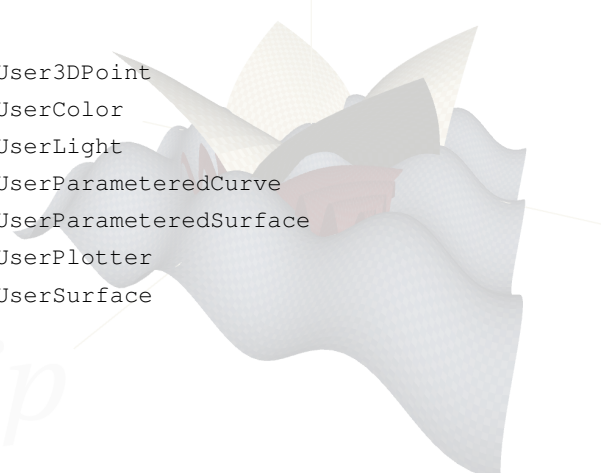
- 
- 1 AjpUser3DPoint
 - 2 AjpUserColor
 - 3 AjpUserLight
 - 4 AjpUserParameteredCurve
 - 5 AjpUserParameteredSurface

Ajp

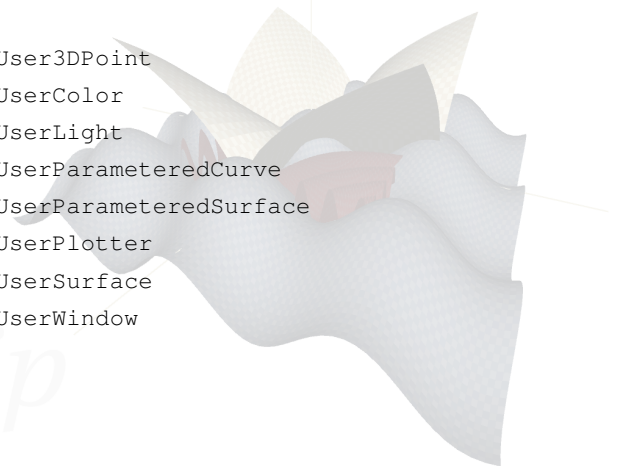
API - Classes

- 
- 1 AjpUser3DPoint
 - 2 AjpUserColor
 - 3 AjpUserLight
 - 4 AjpUserParameteredCurve
 - 5 AjpUserParameteredSurface
 - 6 AjpUserPlotter

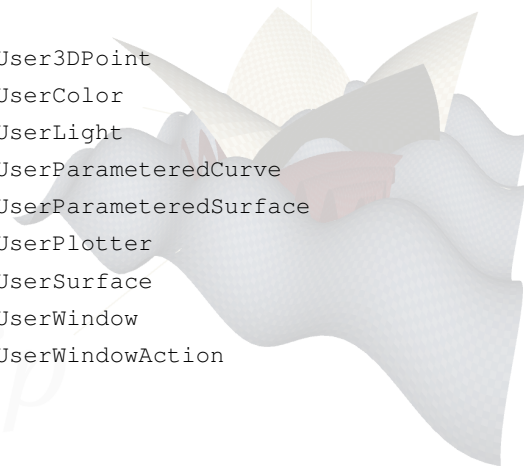
API - Classes

- 
- 1 AjpUser3DPoint
 - 2 AjpUserColor
 - 3 AjpUserLight
 - 4 AjpUserParameteredCurve
 - 5 AjpUserParameteredSurface
 - 6 AjpUserPlotter
 - 7 AjpUserSurface

API - Classes

- 
- 1 AjpUser3DPoint
 - 2 AjpUserColor
 - 3 AjpUserLight
 - 4 AjpUserParameteredCurve
 - 5 AjpUserParameteredSurface
 - 6 AjpUserPlotter
 - 7 AjpUserSurface
 - 8 AjpUserWindow

API - Classes

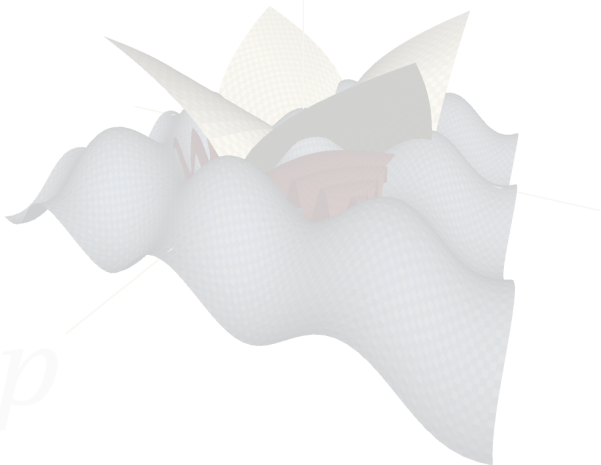
- 
- 1 AjpUser3DPoint
 - 2 AjpUserColor
 - 3 AjpUserLight
 - 4 AjpUserParameteredCurve
 - 5 AjpUserParameteredSurface
 - 6 AjpUserPlotter
 - 7 AjpUserSurface
 - 8 AjpUserWindow
 - 9 AjpUserWindowAction

API - AjpUserScript

```
package org.lozi.ajp.ajp.scripts;  
import org.lozi.ajp.ajp.*;  
  
public interface AjpUserScript  
{  
    /** The main script function which is run when the script is executed  
        .*/  
    public abstract void start( AjpUserPlotter plotter );  
}
```

Listing 1 – L'interface AjpUserScript

API - AjpUserScript



API - AjpUserScript

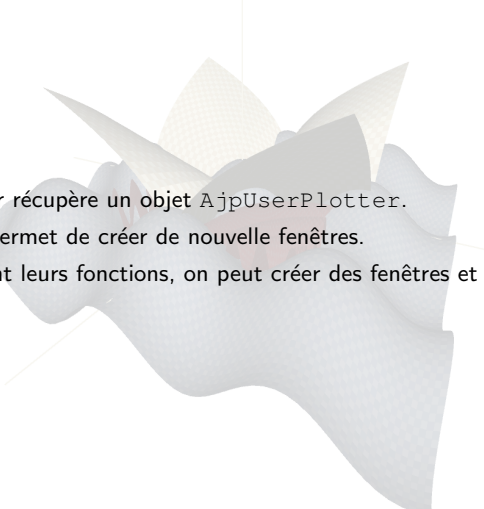
- L'utilisateur récupère un objet AjpUserPlotter.

Ajp

API - AjpUserScript

- L'utilisateur récupère un objet `AjpUserPlotter`.
- Cet objet permet de créer de nouvelles fenêtres.

API - AjpUserScript

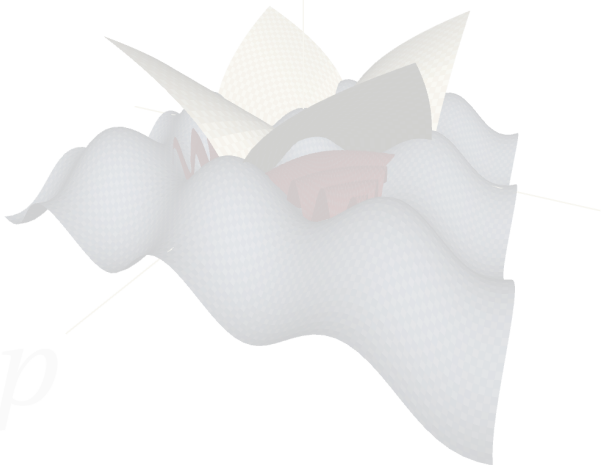
- 
- L'utilisateur récupère un objet `AjpUserPlotter`.
 - Cet objet permet de créer de nouvelle fenêtres.
 - En appelant leurs fonctions, on peut créer des fenêtres et des courbes.

API - AjpPlotter

```
/**
 * Creates a new window.
 *
 * @param title The window title.
 * @param width The window width.
 * @param height The window height.
 * @param x The window x position.
 * @param y The window y position.
 * @return The newly created window.
 */
public AjpUserWindow newWindow( String title, int width, int height,
    int x, int y, AjpUserLight[] lightsArray )
```

Listing 2 – La fonction newWindow

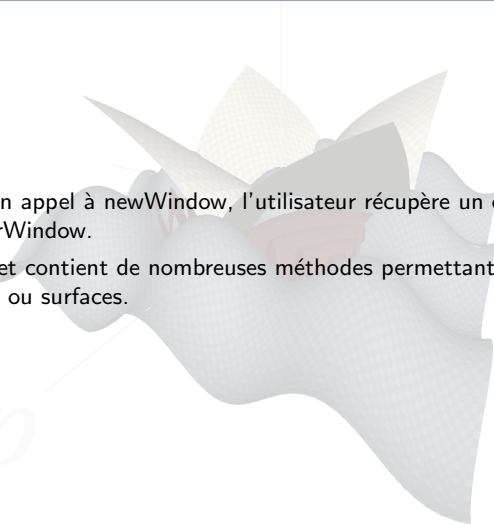
API - AjpUserWindow



API - AjpUserWindow

- Après un appel à `newWindow`, l'utilisateur récupère un objet `AjpUserWindow`.

API - AjpUserWindow

- 
- Après un appel à `newWindow`, l'utilisateur récupère un objet `AjpUserWindow`.
 - Cet objet contient de nombreuses méthodes permettant d'ajouter des courbes ou surfaces.

API - AjpUserWindow (1)

```
/* Fonctions publiques de la classe AjpUserWindow. */  
public void animate ()  
public ArrayList<AjpUserParameteredCurve>  
    getParameteredCurvesArrayList ()  
public ArrayList<AjpUserParameteredSurface>  
    getParameteredSurfacesArrayList ()  
public ArrayList<AjpUserSurface> getSurfacesArrayList ()  
public void setRotationSteps( double leftRightStep, double  
    topBottomStep )  
public void setAxesBounds( double xLowerBound, double xUpperBound,  
    double yLowerBound, double yUpperBound,  
    double zLowerBound, double zUpperBound )  
public void setAxesColor( AjpUserColor color )  
public void setVisible( boolean visibility )  
public void setBackgroundColor( AjpUserColor color )
```

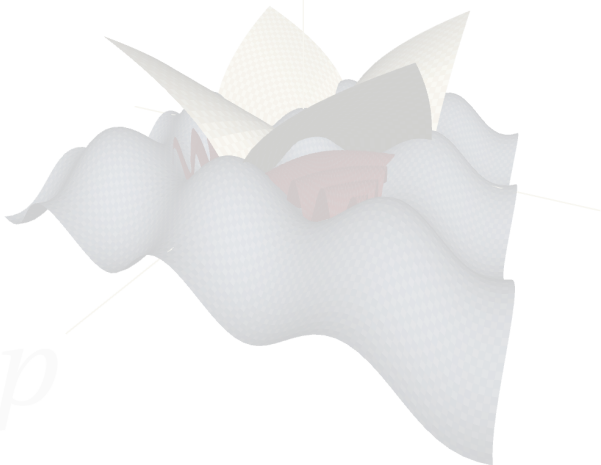
Listing 3 – Liste des fonctions (1)

API - AjpUserWindow (2)

```
public void setParameteredCurvesArrayList( ArrayList<
    AjpUserParameteredCurve> parameteredCurvesArrayList )
public void setParameteredSurfacesArrayList( ArrayList<
    AjpUserParameteredSurface> parameteredSurfacesArrayList )
public void setSurfacesArrayList( ArrayList<AjpUserSurface>
    surfacesArrayList )
public void addParameteredCurve( AjpUserParameteredCurve curve )
public void addParameteredSurface( AjpUserParameteredSurface surface
    )
public void addSurface( AjpUserSurface surface )
public void setInitialCameraPositionAndDirection( AjpUser3DPoint
    position, double xRot, double yRot, double zRot )
```

Listing 4 – Liste des fonctions (2)

API - AjpUserWindow



API - AjpUserWindow

- On peut associer une action à une fenêtre.

Ajp

API - AjpUserWindow

- On peut associer une action à une fenêtre.
- Celle-ci dérive de la classe AjpUserWindowAction.

API - AjpUserWindow

- On peut associer une action à une fenêtre.
- Celle-ci dérive de la classe AjpUserWindowAction.
- Une seule méthode, `run` qui recoit une AjpUserWindow.

API - AjpUserWindow

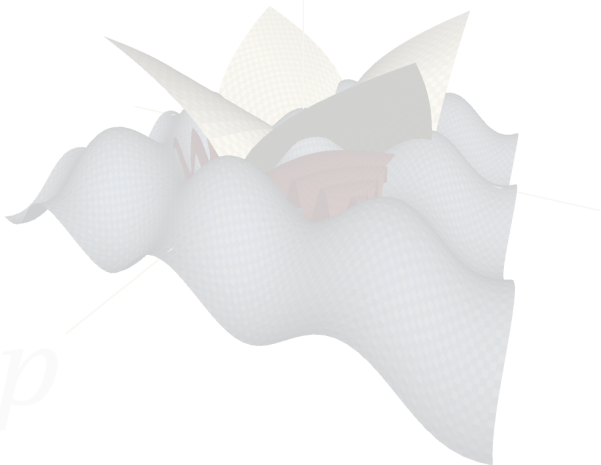
- On peut associer une action à une fenêtre.
- Celle-ci dérive de la classe AjpUserWindowAction.
- Une seule méthode, `run` qui reçoit une AjpUserWindow.
- Un appel par mise à jour de l'affichage (dépendant de la puissance de la machine hôte).

API - AjpUserWindow

```
/**  
 * Set the window's action.  
 *  
 * @param action The action to bind.  
 */  
public void setAction( AjpUserWindowAction action )
```

Listing 5 – La fonction setAction

API - AjpUserWindow

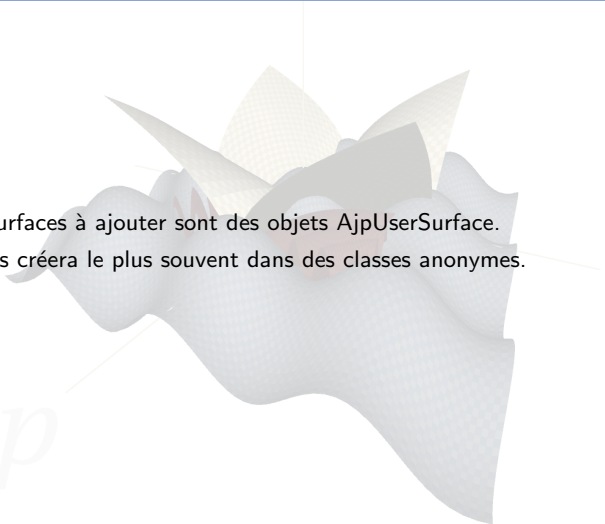


API - AjpUserWindow

- Les surfaces à ajouter sont des objets AjpUserSurface.

Ajp

API - AjpUserWindow

- 
- Les surfaces à ajouter sont des objets AjpUserSurface.
 - On les créera le plus souvent dans des classes anonymes.

API - AjpUserSurface

```
public abstract class AjpUserSurface {  
    /** The x lower bound. */  
    protected double xLowerBound;  
    /** The x upper bound. */  
    protected double xUpperBound;  
    /** The y lower bound. */  
    protected double yLowerBound;  
    /** The y upper bound. */  
    protected double yUpperBound;  
    /** The grid's size. */  
    protected double gridSize;  
    /** The plot's color. */  
    protected String colorString;
```

Listing 6 – La classe AjpUserSurface (1)

API - AjpUserSurface

```
/**  
 * Evaluates the curve's function at the (x,y) position  
 *  
 * @param t The parameter for which the function should be evaluated.  
 * @return The evaluation.  
 */  
public abstract double evaluate( double x, double y );
```

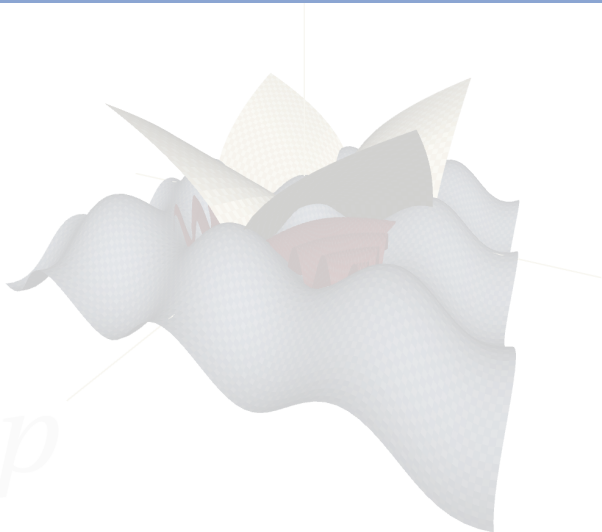
Listing 7 – La classe AjpUserSurface (2)

API - AjpUserSurface

```
/**  
 * Initializes the local variables.  
 */  
public abstract void init ();
```

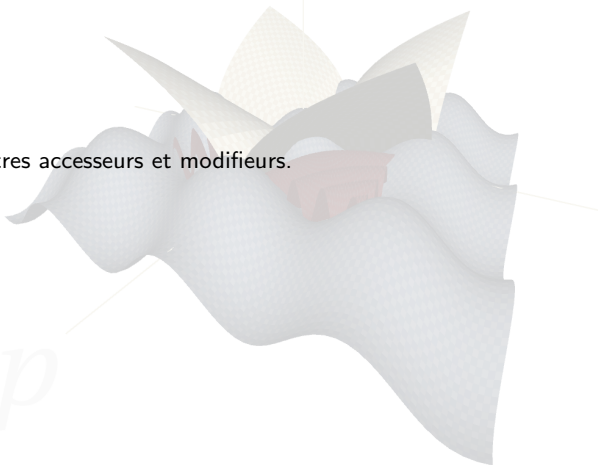
Listing 8 – La classe AjpUserSurface (3)

API



API

- D'autres accesseurs et modifieurs.



API

- D'autres accesseurs et modifieurs.
- Exemple de création de fenêtre et d'ajout de courbe.

Ajp

API - Exemple

```
AjpUserWindow window = plotter.newWindow( "First test window", 320,  
      240, 10, 10, lights );  
window.addSurface( new AjpUserSurface() {  
    public double evaluate( double x, double y ) {  
    return Math.cos( x ) + Math.cos( y );  
    }  
}
```

Listing 9 – Un exemple (1)

API - Exemple

```
public void init() {  
this.colorString = "Blue";  
this.xLowerBound = -10;  
this.xUpperBound = 10;  
this.yLowerBound = -10;  
this.yUpperBound = 10;  
this.gridSize = 0.3;  
}  
});
```

Listing 10 – Un exemple (2)

API - AjpUserParameteredXxx

- Les courbes paramétrées se font à l'aide de la classe `AjpUserParameteredCurve`

API - AjpUserParameteredXxx

- Les courbes paramétrées se font à l'aide de la classe `AjpUserParameteredCurve`
- Les surfaces paramétrées se font à l'aide de la classe `AjpUserParameteredSurface`

API - AjpUserParameteredCurve

```
public abstract class AjpUserParameteredCurve {  
    /** The lower bound for the parameter t. */  
    protected double tLowerBound;  
    /** The upper bound for the parameter t. */  
    protected double tUpperBound;  
    /** The plot color. */  
    protected AjpUserColor color;  
    /** The line width. */  
    protected float lineWidth;  
    /** The step. */  
    protected double step;
```

Listing 11 – La classe AjpUserParameteredCurve (1)

API - AjpUserParameteredCurve

```
/**  
 * Evaluates the function for the parameters s and t.  
 * Should return a tridimensionnal array of double values.  
 *  
 * @param s The first parameter for which the function should be  
 *         evaluated.  
 * @param t The scond parameter for which the function should be  
 *         evaluated.  
 * @return The evaluated point.  
 */  
public abstract AjpUser3DPoint evaluate( double s, double t );
```

Listing 12 – La classe AjpUserParameteredCurve (2)

API - AjpUserParameteredCurve

```
/**  
 * Initializes the local variables.  
 */  
public abstract void init ();
```

Listing 13 – La classe AjpUserParameteredCurve (3)

API - AjpUserParameteredSurface

```
public abstract class AjpUserParameteredSurface {  
    /** The lower bound for the parameter t. */  
    protected double tLowerBound;  
    /** The upper bound for the parameter t. */  
    protected double tUpperBound;  
    /** The lower bound for the parameter s. */  
    protected double sLowerBound;  
    /** The upper bound for the parameter s. */  
    protected double sUpperBound;  
    /** The step for the s parameter. */  
    protected float sStep;  
    /** The step for the t parameter. */  
    protected float tStep;  
    /** The plot's color. */  
    protected String colorString;  
}
```

Listing 14 – La classe AjpUserParameteredSurface (1)

API - AjpUserParameteredSurface

```
/**  
 * Evaluates the function for the parameters s and t.  
 * Should return a tridimensionnal array of double values.  
 *  
 * @param s The first parameter for which the function should be  
 *         evaluated.  
 * @param t The scond parameter for which the function should be  
 *         evaluated.  
 * @return The evaluated point.  
 */  
public abstract AjpUser3DPoint evaluate( double s, double t );
```

Listing 15 – La classe AjpUserParameteredSurface (2)

API - AjpUserParameteredSurface

```
/**  
 * Initializes the local variables.  
 */  
public abstract void init ();
```

Listing 16 – La classe AjpUserParameteredSurface(3)

API - AjpUserParameteredSurface

- D'autres accesseurs et modifieurs.



API - AjpUserParameteredSurface

- D'autres accesseurs et modifieurs.
- Exemple d'ajout de courbe paramétrée.

Ajp

API Exemple

```
window.addOneParameterCurve( new AjpUserParameteredCurve() {  
    public AjpUser3DPoint evaluate( double t ) {  
        double R=1.0;  
        double r=0.2;  
        double n=17.0/2;  
        return new AjpUser3DPoint( ( R + r * Math.cos( n * t ) ) * Math.cos( t  
            ) * 15,  
            ( R + r * Math.cos( n * t ) ) * Math.sin( t ) * 15,  
            ( r * Math.sin( n * t ) * 15 ) );  
    }  
}
```

Listing 17 – AjpUserParameteredSurface - Un exemple (1)

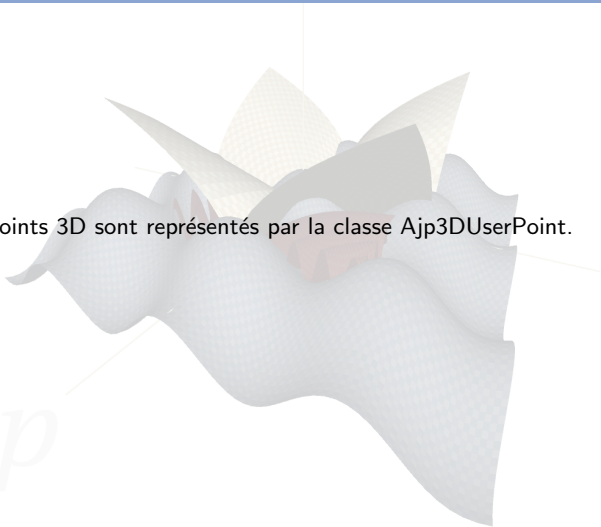
API - Exemple

```
public void init() {  
    this.color = new AjpUserColor( .9f, 0.8f, 0.9f, 1.0f );  
    this.tLowerBound = -7;  
    this.tUpperBound = 7;  
    this.step = 0.01;  
}  
});
```

Listing 18 – AjpUserParameteredSurface - Un exemple (2)

API - AjpUser3DPoint

- Les points 3D sont représentés par la classe Ajp3DUserPoint.



API - AjpUser3DPoint

- Les points 3D sont représentés par la classe Ajp3DUserPoint.
- Elle contient les 3 coordonnées des courbes.

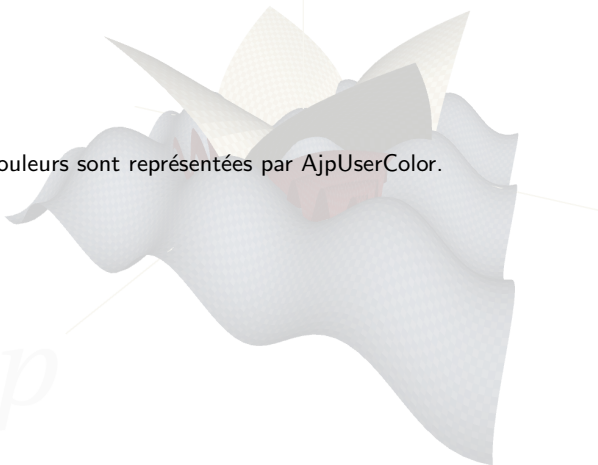
API - AjpUser3DPoint

```
public AjpUser3DPoint( double x, double y, double z )  
public double getX()  
public double getY()  
public double getZ()  
public void setX(double x)  
public void setY(double y)  
public void setZ(double z)
```

Listing 19 – La classe AjpUser3DPoint

API - AjpUser3DPoint

- Les couleurs sont représentées par AjpUserColor.



API - AjpUser3DPoint

- Les couleurs sont représentées par AjpUserColor.
- Elles contiennent les 3 composantes rouge, verte et bleue et le canal alpha.

API - AjpUserColor

```
public AjpUserColor( float red, float green, float blue, float alpha
    )
public float  getRed()
public float  getGreen()
public float  getBlue()
public float  getAlpha()
public void   setRed(float red)
public void   setGreen(float green)
public void   setBlue(float blue)
public void   setAlpha(float alpha)
```

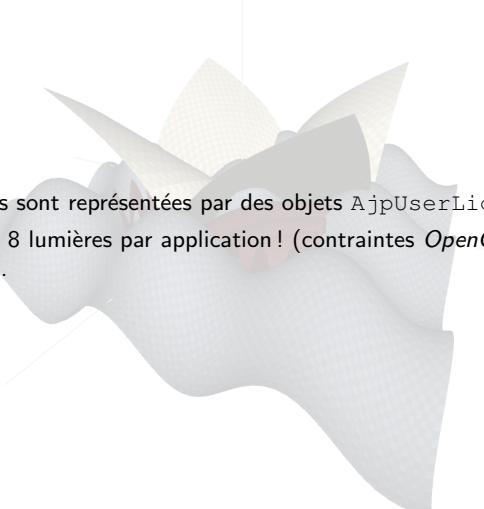
Listing 20 – La classe AjpUserColor

API - AjpUserLight

- Les lumières sont représentées par des objets AjpUserLight.



API - AjpUserLight

- 
- Les lumières sont représentées par des objets `AjpUserLight`.
 - Pas plus de 8 lumières par application ! (contraintes *OpenGL* et matérielles).

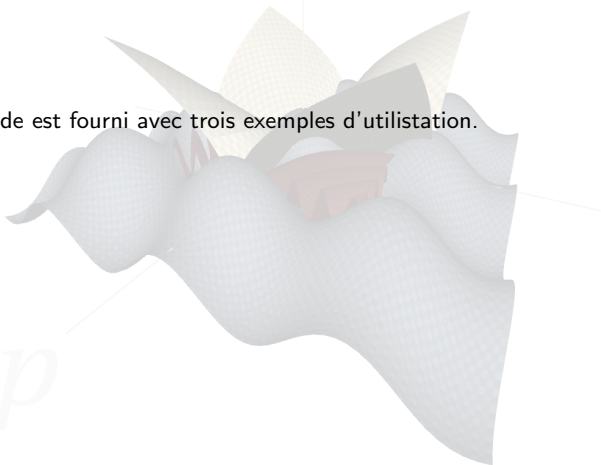
API - AjpUserLight

```
public AjpUserLight ( AjpUser3DPoint position, AjpUserColor color )  
public AjpUserColor getColor()  
public AjpUser3DPoint getPosition()  
public void setColor(AjpUserColor color)  
public void setPosition(AjpUser3DPoint position)
```

Listing 21 – La classe AjpUserLight

Exemples

- Le code est fourni avec trois exemples d'utilisation.



Exemples

- Le code est fourni avec trois exemples d'utilisation.
- Le premier crée deux fenêtres avec des courbes et surfaces diverses.

Ajp

Exemples

- Le code est fourni avec trois exemples d'utilisation.
- Le premier crée deux fenêtres avec des courbes et surfaces diverses.
- Le second code crée une courbe et une surface animées.

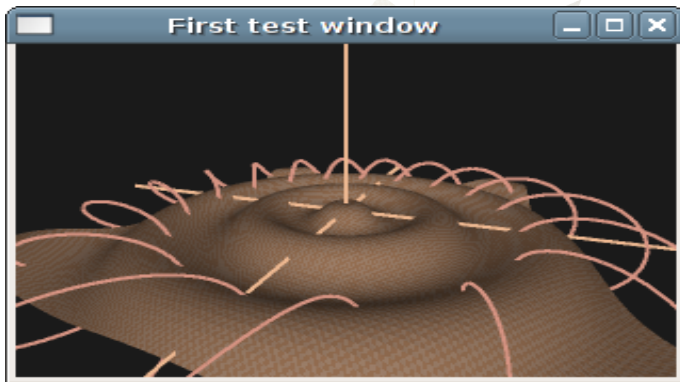
Exemples

- Le code est fourni avec trois exemples d'utilistation.
- Le premier crée deux fenêtres avec des courbes et surfaces diverses.
- Le second code crée une courbe et une surface animées.
- Le troisième crée une surface paramétrée.

Exemples

- Le code est fourni avec trois exemples d'utilistation.
- Le premier crée deux fenêtres avec des courbes et surfaces diverses.
- Le second code crée une courbe et une surface animées.
- Le troisième crée une surface paramétrée.
- Nous allons étudier le second.

Exemples



Exemples

```
package org.lozi.ajp.ajp.scripts;
import org.lozi.ajp.ajp.*;
import java.util.ArrayList;
public class Example2 implements AjpUserScript
{
    public void start( AjpUserPlotter plotter ) {
        // We create the lights.
        AjpUserLight[] lights = new AjpUserLight[] {
            new AjpUserLight( new AjpUser3DPoint( 0, 0, 5
            ),
            new AjpUserColor( .5f, .5f, .5f, 1f ) ) };

        // We create the window.
        AjpUserWindow window = plotter.newWindow( "First test
            window", 320, 240, 10, 10, lights );
        // We set the background color.
        window.setBackgroundColor( new AjpUserColor( 0.1f, 0.1f,
            0.1f, 1.0f) );
    }
}
```

Listing 22 – La classe Example2 (1)

Exemples

```
// We add a surface to the window.  
window.addSurface( new MySurface() );  
// We add a parametered curve.  
window.addParameteredCurve( new MyParameteredCurve() );  
// We set the window's action.  
window.setAction( new AjpUserWindowAction() {  
    public void run ( AjpUserWindow window ) {  
        MySurface surface = (MySurface)window.  
            getSurfacesArrayList().get( 0 );  
        MyParameteredCurve curve = (  
            MyParameteredCurve)window.  
            getParameteredCurvesArrayList().get( 0 );  
        // We increment the parameter T of the first  
            surface.  
        surface.setT( surface.getT() + 0.1 );  
        curve.setU( curve.getU() + 0.01 );  
    }  
});
```

Listing 23 – La classe Example2 (2)

Exemples

```
// We set the rotation steps.  
window.setRotationSteps( 0.1, 0 );  
// We set the axes' bounds.  
window.setAxesBounds( -20, 20, -20, 20, -20, 20 );  
// We animate the window.  
window.animate();  
}  
class MySurface extends AjpUserSurface {  
    // We set a parameter that we can update.  
    private double t;  
  
    public double evaluate( double x, double y ) {  
        return Math.cos( Math.sqrt( x * x + y * y ) + t );  
    }  
}
```

Listing 24 – La classe Example2 (3)

Exemples

```
public void init() {  
    this.colorString = "Orange";  
    this.xLowerBound = -15;  
    this.xUpperBound = 15;  
    this.yLowerBound = -15;  
    this.yUpperBound = 15;  
    this.gridSize = 0.3;  
}  
public double getT() {  
    return t;  
}  
public void setT( double t ) {  
    this.t = t;  
}  
}
```

Listing 25 – La classe Example2 (4)

Exemple

```
class MyParameteredCurve extends AjpUserParameteredCurve {  
    // We set a second parameter.  
    public void init() {  
        this.color = new AjpUserColor( .9f, 0.8f, 0.9f, 1.0  
            f );  
        this.tLowerBound = -7;  
        this.tUpperBound = 7;  
        this.step = 0.01;  
    }  
    public double getU() {  
        return u;  
    }  
    public void setU( double u ) {  
        this.u = u;  
    }  
}
```

Listing 26 – La classe Example2 (5)

Conclusion (1)

- La plus grande partie du projet est fonctionnelle.

Ajp

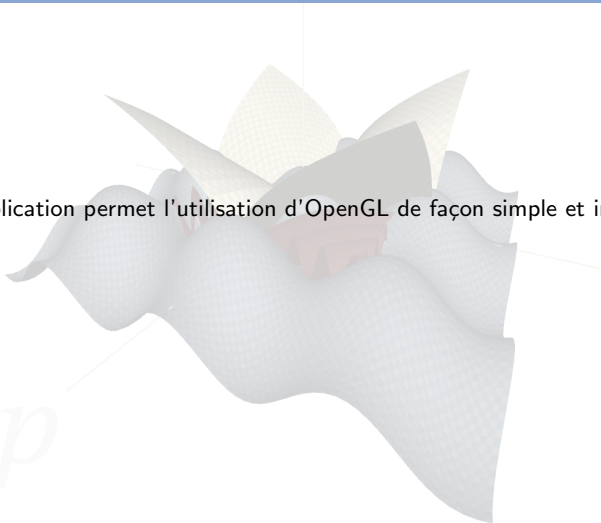
Conclusion (1)

- La plus grande partie du projet est fonctionnelle.
- Il reste à gérer différents types d'objets.

Ajp

Conclusion (2)

- L'application permet l'utilisation d'OpenGL de façon simple et intuitive.



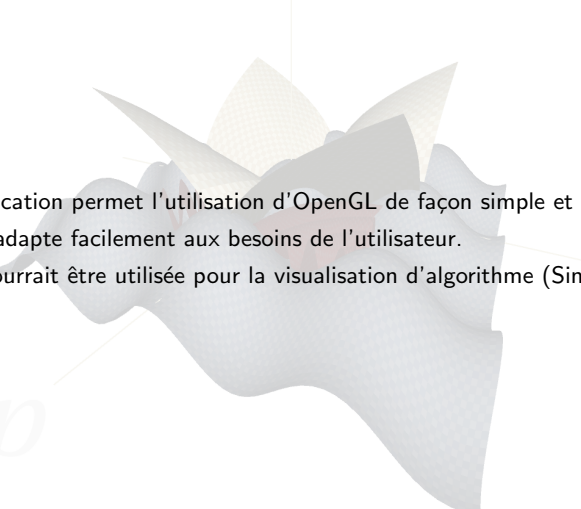
Ajp

Conclusion (2)

- L'application permet l'utilisation d'OpenGL de façon simple et intuitive.
- Elle s'adapte facilement aux besoins de l'utilisateur.

Ajp

Conclusion (2)

- 
- L'application permet l'utilisation d'OpenGL de façon simple et intuitive.
 - Elle s'adapte facilement aux besoins de l'utilisateur.
 - Elle pourrait être utilisée pour la visualisation d'algorithmes (Simplex...).