CEIG – Spanish Computer Graphics Conference (2021)
A. Chica and L. Ortega (Editors)

ISSIGraph: An open source multi-platform C++ tool for rapid 2D/3D wireframe sketching

C. Jiménez de Parga, Ph.D. [1]

[1]Computer Systems and Software Engineering Department (ISSI) - National Distance Education University (UNED)

Abstract

Rapid sketch modelling and Computer-Aided Design (CAD) initiation are increasingly demanded activities in the creative and educational areas. For this reason, this paper presents a cross-platform C++ toolkit with the aim to facilitate the illustration of technical concepts in a fast way using basic quadric objects, Bézier and NURBS surfaces with a wireframe representation. This tool was designed using Software Engineering principles guided by a basic Rational Unified Process (RUP) methodology with the application of the Unified Modelling Language (UML). This tool perfectly works in all computers with very elemental 3D graphics hardware and with OpenGL support. The resulting benchmarks demonstrate that ISSIGraph has a very small CPU footprint that make it suitable for any platform. As a consequence, this application is well suited for rapid 2D/3D project sketching in the creative and engineering fields, as well as an initiation to CAD techniques for students and computer fans.

CCS Concepts

• ***Human-centered computing*** → *Accessibility theory, concepts and paradigms;* • ***Applied computing*** → ***Arts and humanities; Computer-assisted instruction; Education;*** • ***Software and its engineering*** → ***Designing software;*** • ***Computing methodologies*** → ***Computer graphics;***

1. Introduction

Design and imaging is as an essential requirement in the information society: everywhere from product conception to architectural design virtual models are generated to help people see what they will get. The following section presents the state of the art in 3D sketching tools; some of them are very popular. The Materials and Methods section explains the most relevant math techniques used to solve usability issues. A complete list of ISSIGraph features with screenshots is presented. The Results section analyzes the tool from an environmental perspective. Finally, the Conclusions section summarizes all work covered in this paper.

2. State of the art and related works

Fourty years back, applications like AutoCAD (https://www.autodesk.es/products/autocad/overview?term=1-YEAR) made computer-aided design possible. Later, other applications such as SketchUp (https://www.sketchup.com/es) eased 3D design from a practical perpective. In the world of General Public License and open source (GPL), many solutions are available from the Solaris platform to UNIX/Linux ones; such is the case of LibreCAD (https://librecad.org/), which adopts the principles of GNU free software and the whole source code is available to the community. LibreCAD is an example of basic but robust architecture aimed at rapid sketching. Besides this, LibreCAD is a proof that building full-featured software in C++ is not limited to big companies. Table 1 compares the mentioned tools versus commercial solutions.

	Free software	Open-source	Easiness
AutoCAD	No	No	No
SketchUp	No	No	No
LibreCAD	Yes	Yes	No
ISSIGraph	Yes	Yes	Yes

Table 1: *Comparative table of various sketching solutions.*

3. Materials and methods

Following the LibreCAD principles, ISSIGraph (http://issigraph.sourceforge.net) is a free and open software application for small companies and non-for-profit organizations that allows rapid sketching of ideas, planes, text and drawing either in 2D or 3D. The layered software architecture of this open-source solution can be extended or reused to create similar applications from its foundation classes. Therefore, ISSIGraph is not a finished tool but the basis for other developers to build their own tools. It is developed with a basic RUP methodology using UML 2.x [JdP21] and C++ implementation on Linux, Mac OS X and Windows platforms with OpenGL 1.x support.

DOI: 10.2312/ceig.20211358

The main idea behind this application is providing technical universities the building blocks to initiate the students in complex software architecture building as part of their tuition, letting them choose between Java webserver 3-tier applications or graphical solutions.

The main features of ISSIGraph are the following:

- Fast 2D/3D sketching.
- Reusable framework for graphical applications (open source).
- Educational CAD initiation.
- Entertainment tool.

Besides this, ISSIGraph benchmarks a low CPU and GPU usage, achieving high performance and minimal energy consumption.

The ISSIGraph graphics user interface (GUI) follows the metrics of Theo Mandel [Man97] in his three golden rules:

- *Place Users in Control:* The ISSIGraph GUI is intuitive. All graphical objects are interactive. It allows redo/undo in a single cycle and copy/paste.
- *Reduce Users' Memory Load:* The ISSIGraph application demands very low use of memory as demonstrated in the results section.
- *Make the Interface Consistent:* No error prone GUI.

Besides this, the user interface tries to be attractive, easy to use and friendly.

ISSIGraph allows sketching with the following objects from the OpenGL Glew and Freeglut:

- Bézier surfaces
- Nurbs surfaces
- Cylinders
- Cones
- Teapots
- Tetrahedrons
- Octahedrons
- Dodecahedrons
- Icosahedrons
- Rombic dodecahedrons
- Toruses
- Vectorial outline text

3.1. The SpinEasy System

In order to develop a versatile graphic system that allows easy modelling, I have created a system that simulates the artisan way of modelling on a lathe. The basic idea is that the user does not have to bother changing the camera view to assess the correct modelling, because the SpinEasy feature facilitates NURBS and Bézier modelling while rotating these surfaces.

The problem behind the SpinEasy system is how to allow the free movement of the control points with no position jump when the surface is rotated or translated. This is produced when the X,Y mouse coordinates in 2D are added to the (x,y,z) control point coordinates in the model domain. For example, if a X axis α rotation is performed:

$$\begin{aligned} z' &= y sin(\alpha) + z cos(\alpha) \\ y' &= y cos(\alpha) - z sin(\alpha) \\ x' &= x \end{aligned} \tag{1}$$

with Equation 1 we achieve an X-axis rotation. If we now want another rotation over Y axis, we will proceed in this way:

$$\begin{aligned} \sigma &= z' \\ \phi &= x' \\ z' &= \sigma cos(\beta) - \phi sin(\beta) \\ x' &= \sigma sin(\beta) + \phi cos(\beta) \end{aligned} \tag{2}$$

A problem arises if we move (x',y',z') coordinates freely relative to mouse coordinates; for example, $x' = x' + 10$ and $y' = y' + 10$ in OpenGL screen coordinates, the new rotation coordinates will fall out of the new orbit. Therefore, to obtain a correct (x',y',z') coordinates we need to provide a coherent rotation pivot. Figure 1 illustrates this situation:

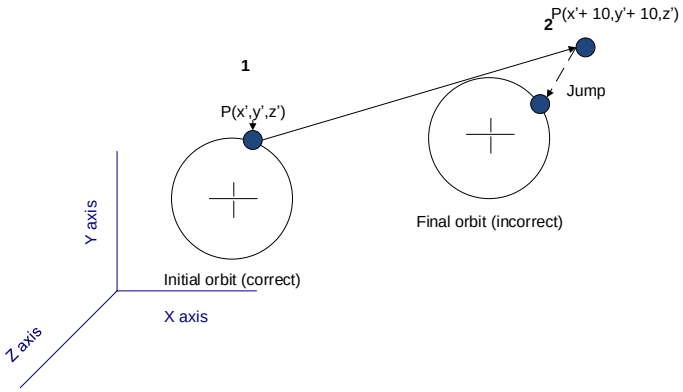

Figure 1: *Stage 1: The moved control point is not correct according to the new orbit translation. In this case an annoying jump is produced when updating the whole control points rotation.*

In order to solve this problem, we must reverse the Equation 1 and 2 as explained in Algorithm 1:

Figure 2 illustrates the final position of the rotation when the mouse-displaced point has found a coherent place inside the orbit.

3.2. Development methodology

The ISSIGraph experimental application has been developed using a simplification of RUP methodology. As an Unified Process (UP) [JdP21], the tasks and iterations are arranged using four fundamental phases:

Algorithm 1: Orbit insertion algorithm: **x, y, z** are the resulting control point positions for the new orbit.
Warning: Does not calculate angle singularities.

$\phi = x' cos(\beta) - z' sin(\beta)$
if $(\beta \neq 0^\circ \wedge \beta \neq 180^\circ)$ **then**
  $\delta = (x' - \phi * cos(\beta))/sin(\beta)$
end
$\sigma = y'$
$\mathbf{x} = \phi$
$\mathbf{z} = \delta cos(\alpha) - \sigma sin(\alpha)$
if $(\alpha \neq 90^\circ \wedge \alpha \neq 270^\circ)$ **then**
  $\mathbf{y} = (\sigma + z sin(\alpha))/cos(\alpha)$
end

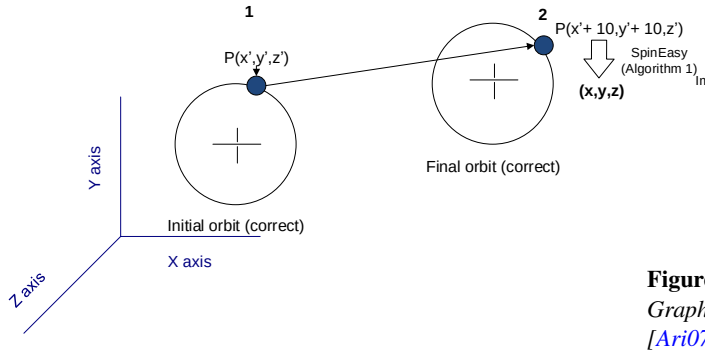

Figure 2: *Stage 2: After calculating Algorithm 1, the moved control point position is inserted in a coherent position around the new rotation orbit.*

- **Inception:** Approximate vision, business analysis, approximated estimations.
- **Elaboration:** Refined vision, iterative implementation of the main architecture core, high-risks resolution, requisite identification, more realistic estimations.
- **Construction:** Iterative implementation of the remaining lower risk requisites and deployment preparation.
- **Transition:** Beta tests, deployment.

UP describes activities such as uses cases and disciplines named *workflows*. In UP an artifact is the common term for any work product: source code, web graphics, diagrams, models, etc.

The four RUP disciplines are the following:

- **Business modelling:** Objects domain model inclusion.
- **Requisites:** Requirements analysis for the application, as use case modelling and non-functional requisites identification.
- **Design:** All design aspects, including global architecture, objects, etc.
- **Implementation:** System programming.

During the first iterations the main effort lays on requisites and design. Towards the end, the major effort lays on implementation. Figure 3 illustrates the ISSIGraph used methodology.

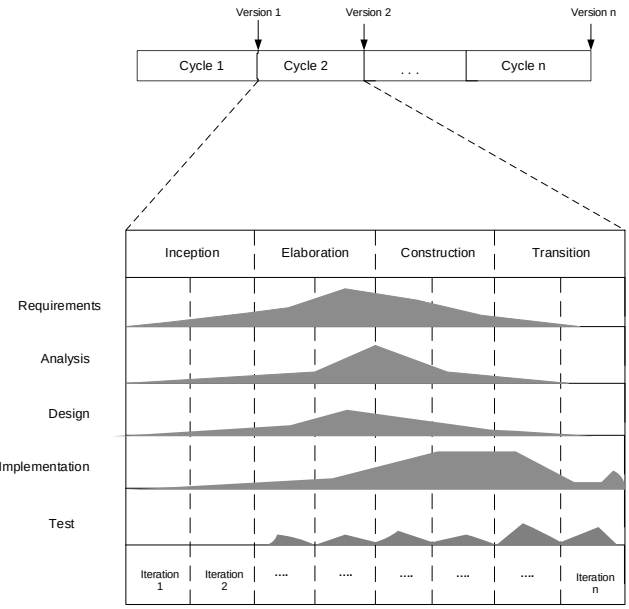

Figure 3: *The simplified RUP methodology applied in the ISSIGraph application with its four phases and its four disciplines [Ari07].*

The UML 2.x artifacts used during development are *Domain model*, *Use case model*, *Class diagram* as static model views, *Sequence diagrams* as dynamic model view.

The applied life cycle is the proper of an UP process, which is an iterative-incremental for one year project duration with possible requirements changes, risk reduction and progressive validation, due to developer's previous experience with trained technology [Cue03]. At a final stage, beta tests were performed for white box ones (sentence and conditionals covering, value limits, etc.), as well as black box tests (stress, usability, security, integrity and performance test).

3.3. Architecture

The ISSIGraph application follows an Object Oriented architecture with a 3-tier organization. The first tier performs the presentation layer with the use of wxWidgets framework (https://www.wxwidgets.org/); the second tier performs business logic layer (geometry and control logic), while the third tier is the Hardware Abstraction Layer (HAL) which manages OpenGL libraries. Figure 4 explains the ISSIGraph architecture layout.

4. Results

The ISSIGraph benckmarks have been performed in the Ubuntu-Linux 18.04 64-bit version using an Intel(R) Core(TM) i7 CPU 860@2.80GHz, 4 physical/8 logic cores, 2.79 GHz (first generation

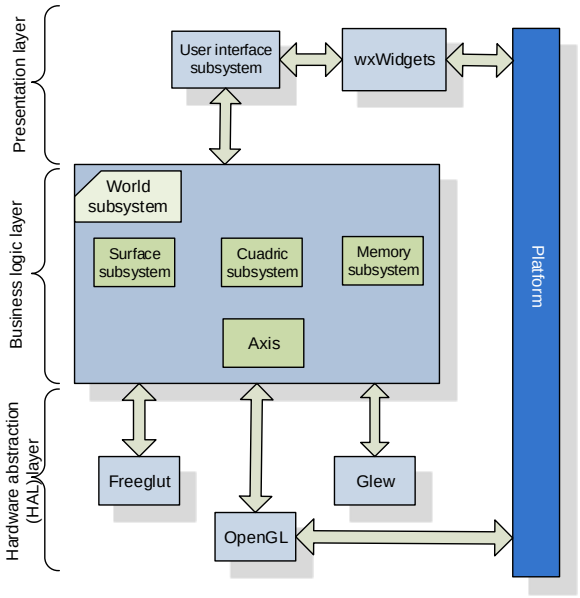

Figure 4: *ISSIGraph architecture exemplification with the 3-tiers: Presentation, Business Logic and Hardware abstraction layers.*

/2009) with 12 GB RAM. The graphics card is an nVidia GTX 1070 non-Ti. The following dataset Table 2 was obtained:

	CPU (max.)	GPU (max.)	Mem MB (max.)	Energy
Execution	4%	15%	15.3	Low
Idle	0%	0%	12.5	Very low

Table 2: *ISSIGraph peformance table.*

The maximum mode was reached when rotating a NURB in full HD (1920 × 1080) in a plentyful scene filled with the following objects: 1 NURB with 32 control points, 10 Toroids, 7 Cones, 5 Spheres, 2 Teapots, outline text with 12 characters long and 3 Dodecahedrons.

5. Conclusions

According to the obtained results and keeping in mind that this project remains in an experimental stage, we can state that this application is suited as a good prototype for CAD initiation for amateur programmers and educational projects, as well as demonstrator for complex software architecture courses in the academic domain.

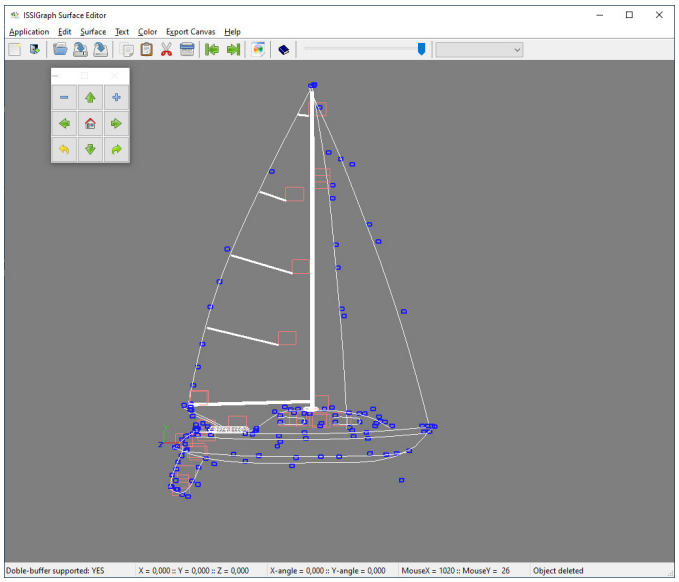

Figure 5: *ISSIGraph 2D sailboat modelling with unfinished NURBS, one torus and cylinders.*

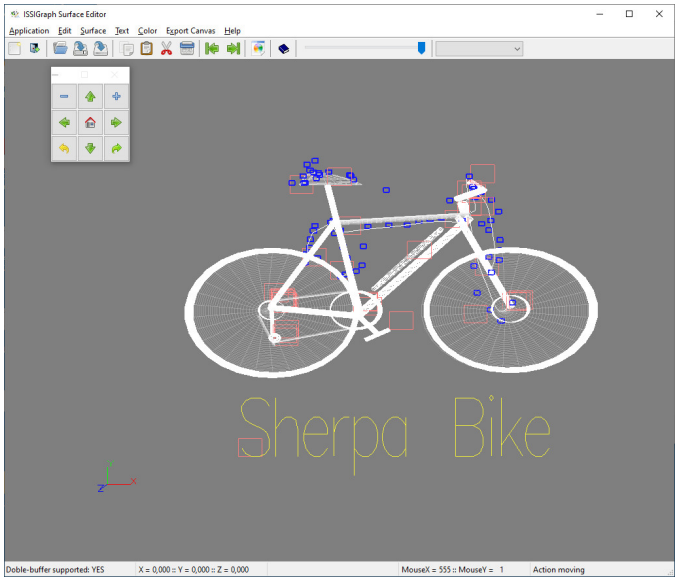

Figure 6: *ISSIGraph 2D bike sketch modelling with one NURB, two cones, cylinders and outline text.*

References

[Ari07] ARIAS M.: *Addenda de la asignatura de Análisis, Diseño y Mantenimiento del Software*, vol. 1. UNED, 2007. 3

[Cue03] CUEVAS G.: *Gestión del Proceso Software*, vol. 1. Editorial Ramón Areces, 2003. 3

[JdP21] JIMÉNEZ DE PARGA C.: *UML: Arquitectura de aplicaciones en Java, C++ y Python*, vol. 2. Editorial Ra-ma, 2021. 1, 2

[Man97] MANDEL T.: *The elements of user interface design*, vol. 20. Wiley New York, 1997. 2