

GraphicsLab 系统使用说明书

Rainorangelemon, rainorangelemon@gmail.com

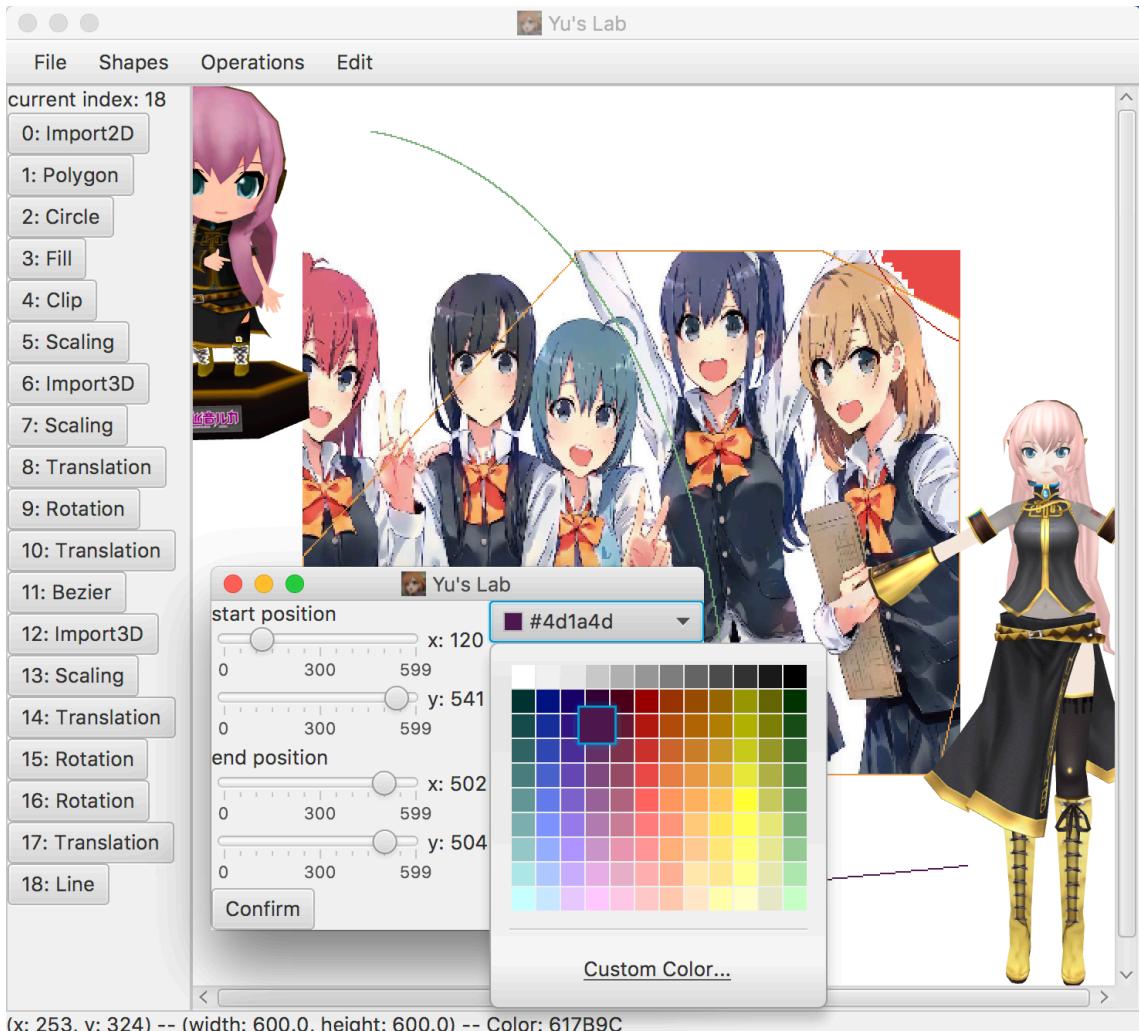
2018 年 6 月 24 日

目录

1 综述	2
2 开发环境	3
3 功能与使用说明	4
3.1 进入系统	4
3.2 图形数据: 输入	5
3.2.1 直线	5
3.2.2 圆/椭圆	6
3.2.3 多边形	7
3.2.4 Bezier 曲线	9
3.2.5 填充	10
3.3 图片: 导入与导出	11
3.3.1 导出图片	11
3.3.2 导入图片	12
3.4 3D 文件: 导入	13
3.4.1 3D 文件: 导入 OFF 文件	13
3.4.2 3D 文件: 导入 OBJ 文件	15
3.5 图形数据: 裁剪	17
3.6 图形数据: 变换	18
3.6.1 变换: 缩放	18
3.6.2 变换: 旋转	19
3.6.3 变换: 平移	20
3.6.4 变换: 连续操作	20
3.6.5 变换: 所支持不同图元的不同变换	21
3.7 图形 & 操作数据: 编辑	22
3.8 撤销与恢复	23
3.8.1 撤销	23
3.8.2 恢复	23

4 系统介绍	24
4.1 目录树与模块代码介绍	24
4.2 关于 License 的说明	25
5 总结	26

1 综述



很顺利，我完成了实验的所有要求，并额外完成了以下功能：

- 填充算法的灵活性：可选择四连通/八连通，边界填充/泛滥填充。
- 对 3D 物体的平移、旋转、缩放操作：平移和旋转操作都是三维操作，缩放是 xy 平面内的缩放。
- 撤销、恢复功能。
- 圆/椭圆的灵活性：可选择中点算法/Bresenham 算法。
- 导入图片的灵活性：可选择 PNG/JPG 格式。
- 导入 3D 模型的灵活性：可选择 OFF 格式，以及更复杂更漂亮的 OBJ 格式。

2 开发环境

- 报告使用 LaTex 编写。
- 开发环境为 MacOS 下的 Eclipse LUNA， Java 版本为 Java 8 Update 171。
- 代码编译说明：请使用 Eclipse LUNA 进行编译，如安装 JavaFX 遇到困难，可自行上网搜索，也可联系本人。
- 代码运行说明：Runnable.jar 是打包版本。如需使用，则双击打开即可，若双击无法打开，请输入命令：java -jar 目标文件.jar。前提是您的电脑中已安装过 Java。
- Resources 中包含两个本人利用 python 脚本修改过的 obj 文件，能够显示在画板上，建议使用它们测试 3D 导入的功能。

3 功能与使用说明

以下介绍如何使用该系统，制作出漂亮的图片。

3.1 进入系统

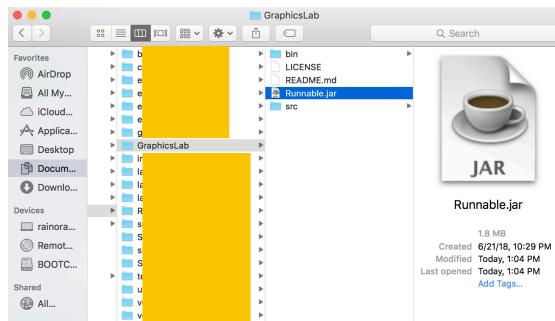


图 1: Runnable.jar 是打包版本。如需测试，则双击打开即可，若双击无法打开，请输入命令：java -jar 目标文件.jar。

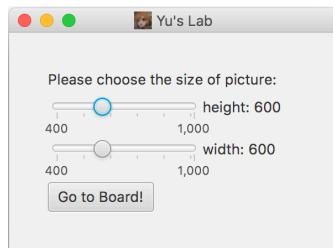


图 2: 选择画板的大小，默认为 600 x 600。

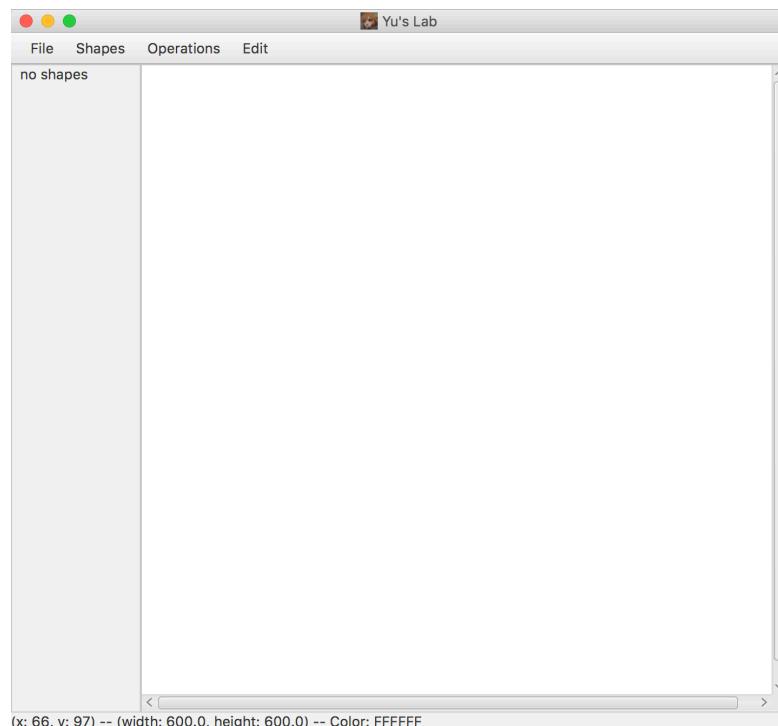


图 3: 成功进入系统。顶部为菜单，也是除编辑操作以外的，所有操作的发源地。左边为编辑栏，针对每次操作，会生成不同的编辑按钮；现在没有任何操作，因此编辑栏没有按钮。

3.2 图形数据：输入

以下介绍如何输入图元。

3.2.1 直线



图 4：在顶部菜单中选择：Shapes→Line

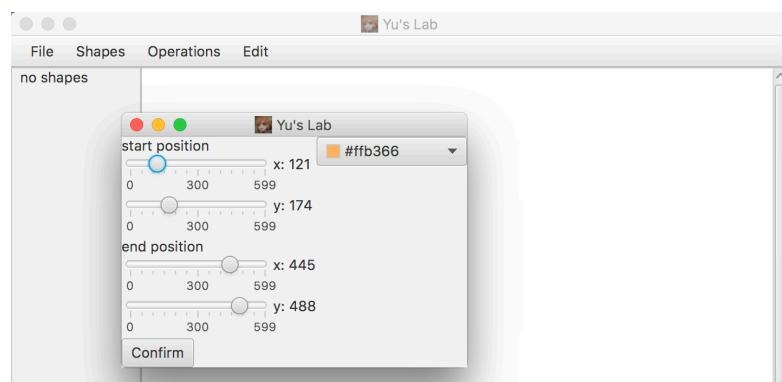


图 5：调整直线的起点与终点，以及直线的颜色

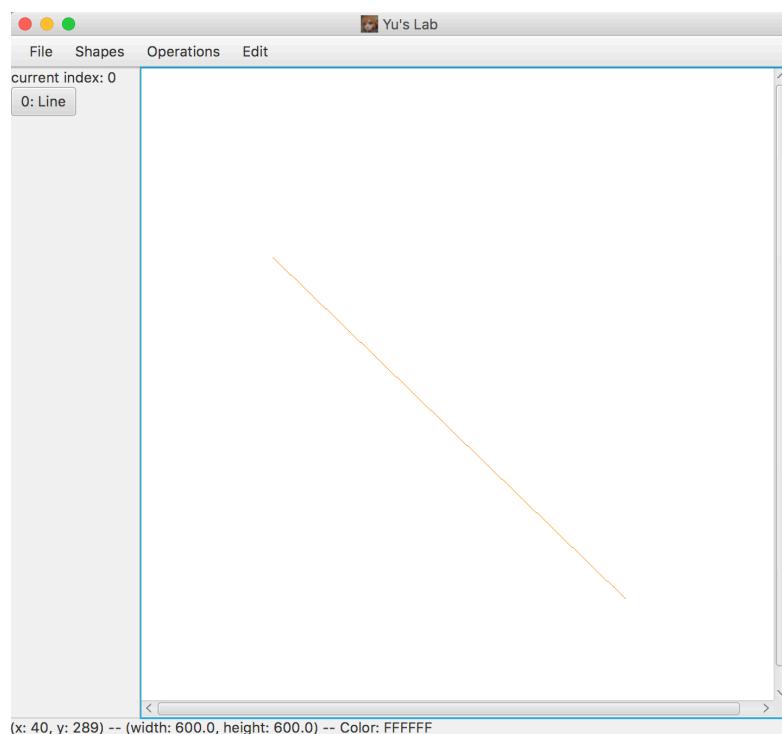


图 6：成功画出直线

3.2.2 圆/椭圆



图 7: 在顶部菜单中选择: Shapes→Circle/Oval

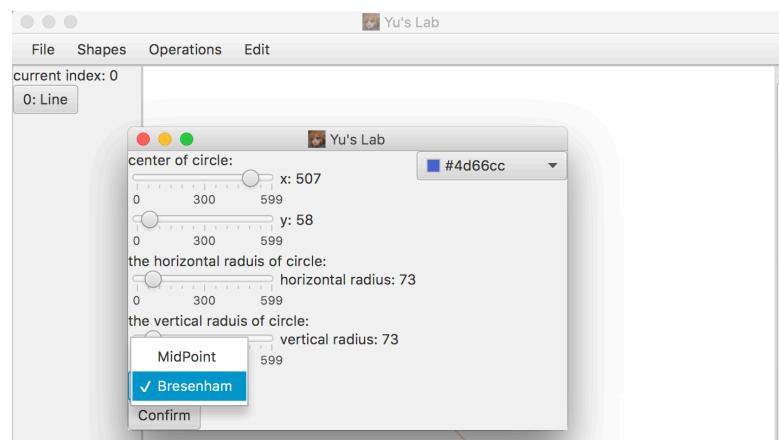


图 8: 调整圆心位置, 以及水平半径与垂直半径。两个半径一致为圆; 否则为椭圆。此外, 也可选择使用中点算法生成还是使用 Bresenham 算法生成。

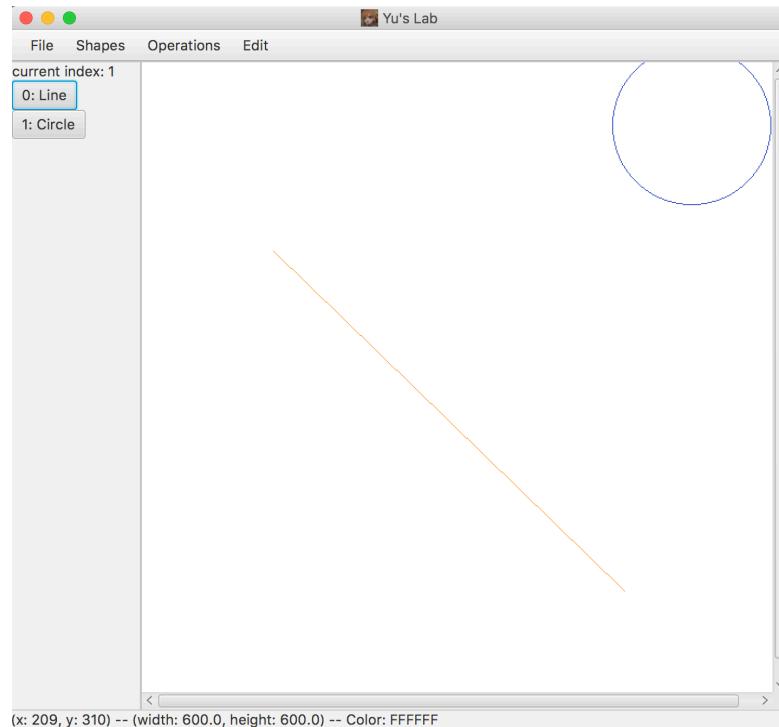


图 9: 成功画出一个圆。

3.2.3 多边形



图 10: 在顶部菜单中选择: Shapes→Polygon

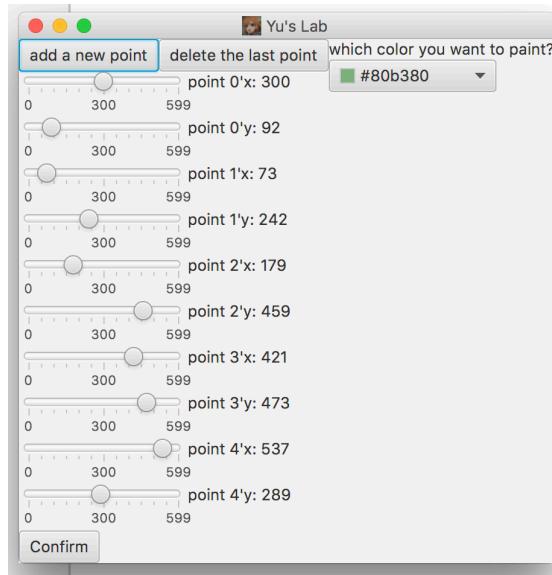


图 11: 按逆时针或顺时针顺序, 添加或删除顶点, 并选择多边形颜色。

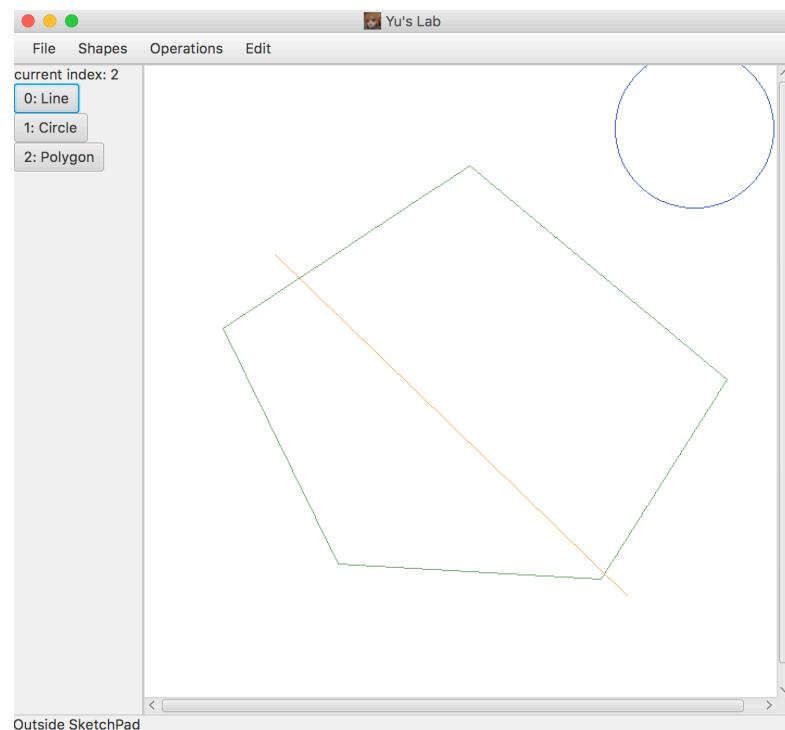


图 12: 成功画出一个多边形。

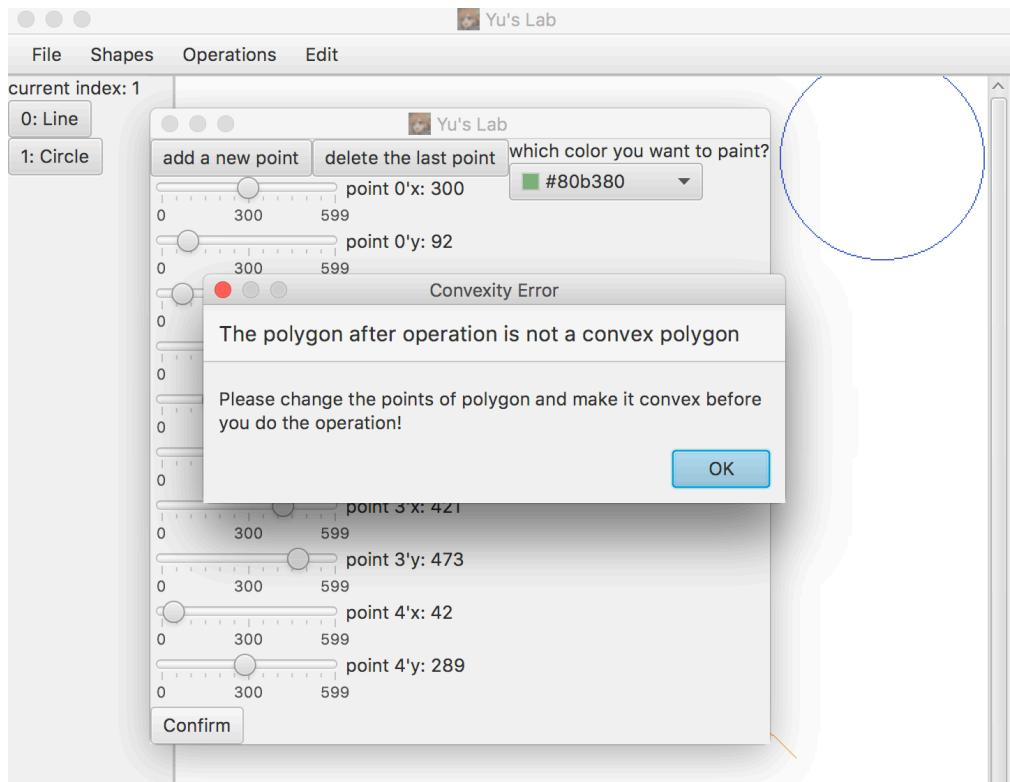


图 13: 如果输入为非凸多边形，则会跳出该弹窗，提示用户修改多边形顶点至多边形为凸，否则不输出

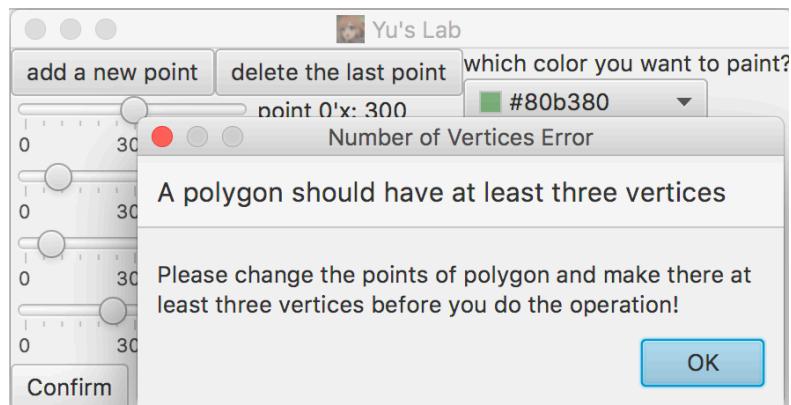


图 14: 如果输入顶点少于三个，则会跳出该弹窗，提示用户添加多边形顶点至多边形顶点大于两个，否则不输出

3.2.4 Bezier 曲线

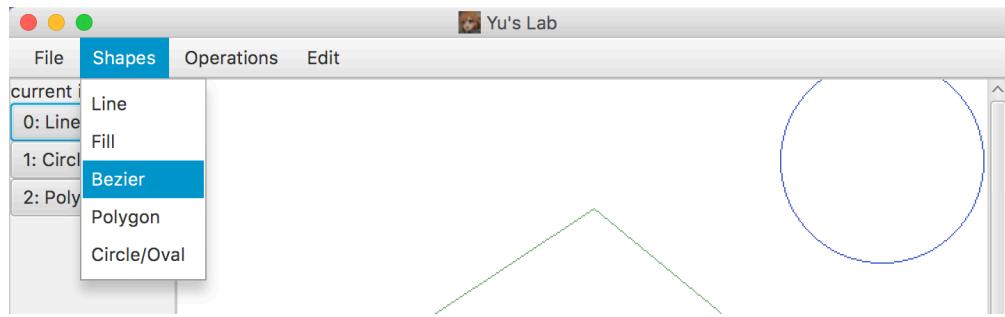


图 15: 在顶部菜单中选择: Shapes→Bezier

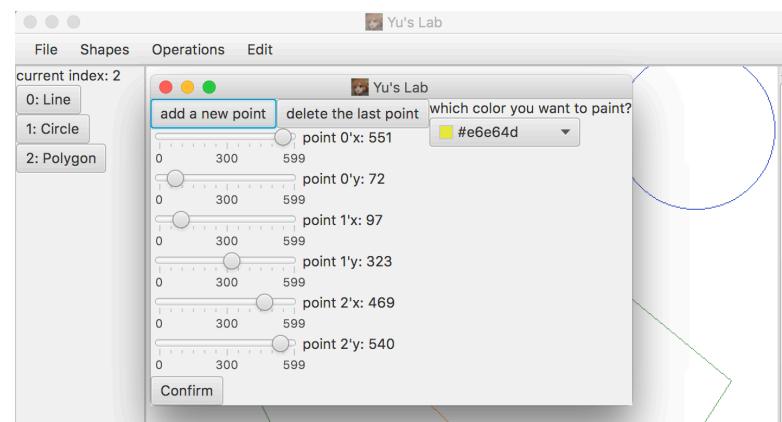


图 16: 按顺序添加或删除控制点，并选择曲线颜色。

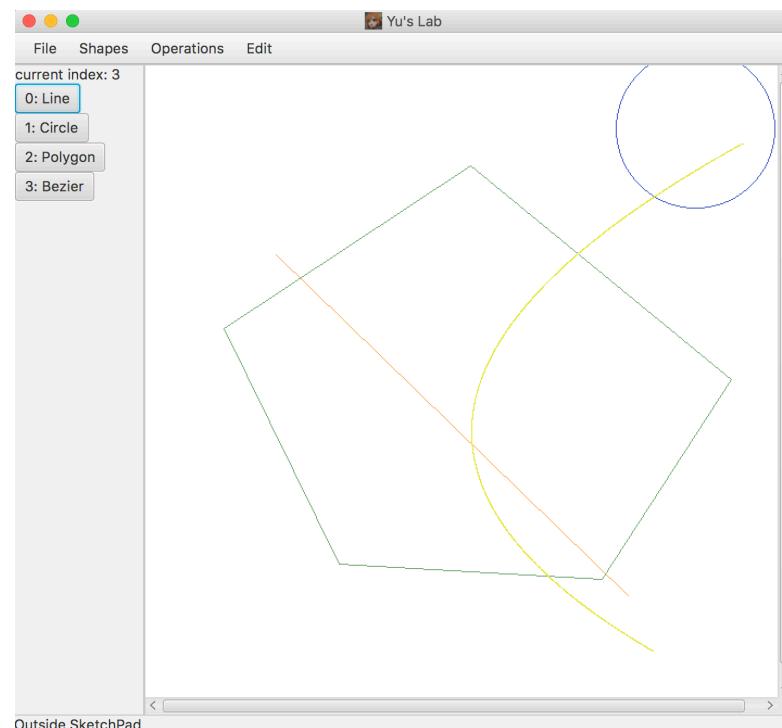


图 17: 成功画出一条 Bezier 曲线。

3.2.5 填充

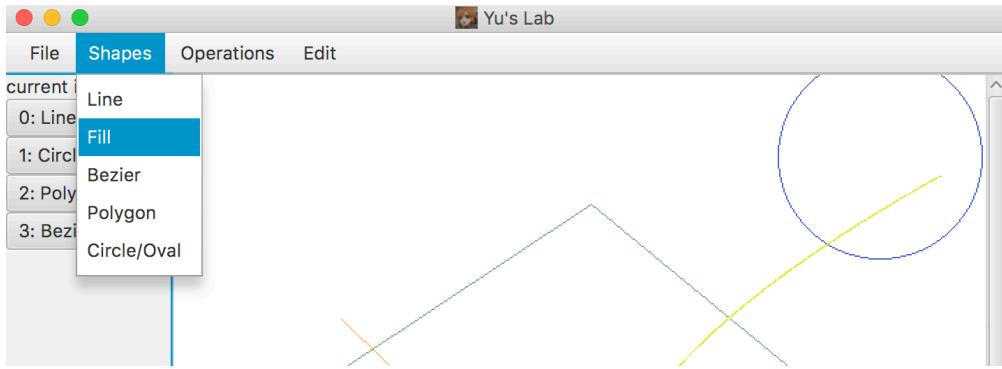


图 18: 在顶部菜单中选择: Shapes→Fill

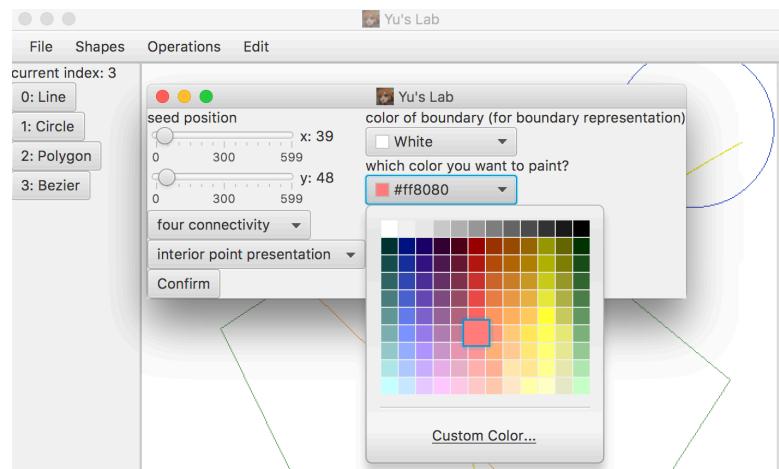


图 19: 选择填充种子点，并选择填充颜色。可选择区域表示为内点表示，还是边界表示。可选择邻居为四连通，还是八连通的邻居。此外，若为边界表示，则需输入边界颜色。

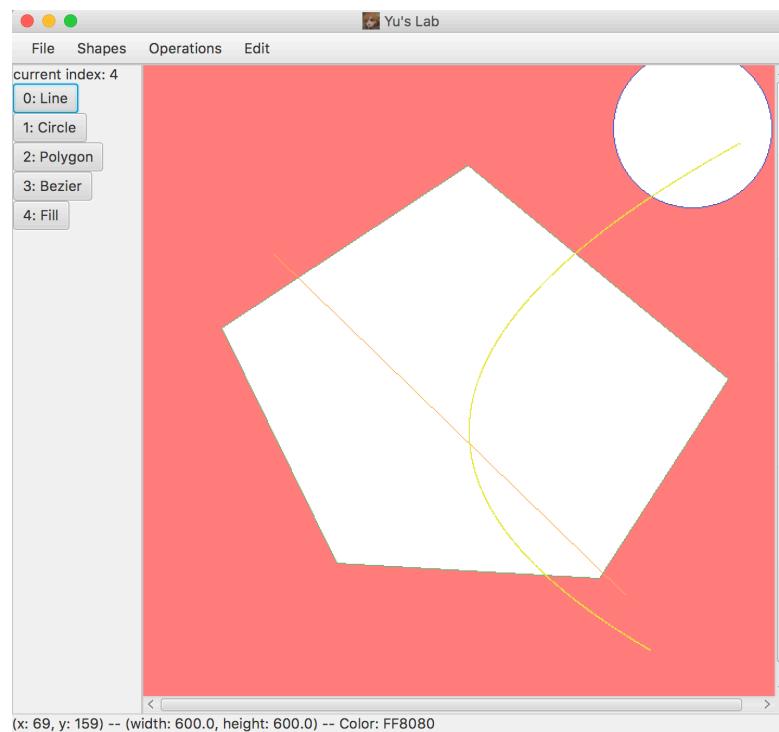


图 20: 成功填充。

3.3 图片：导入与导出

3.3.1 导出图片

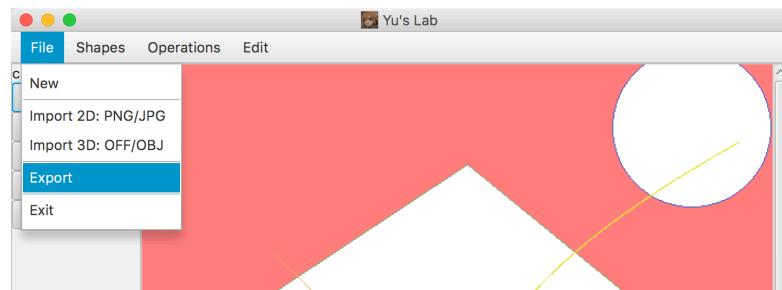


图 21: 在顶部菜单中选择: File→Export

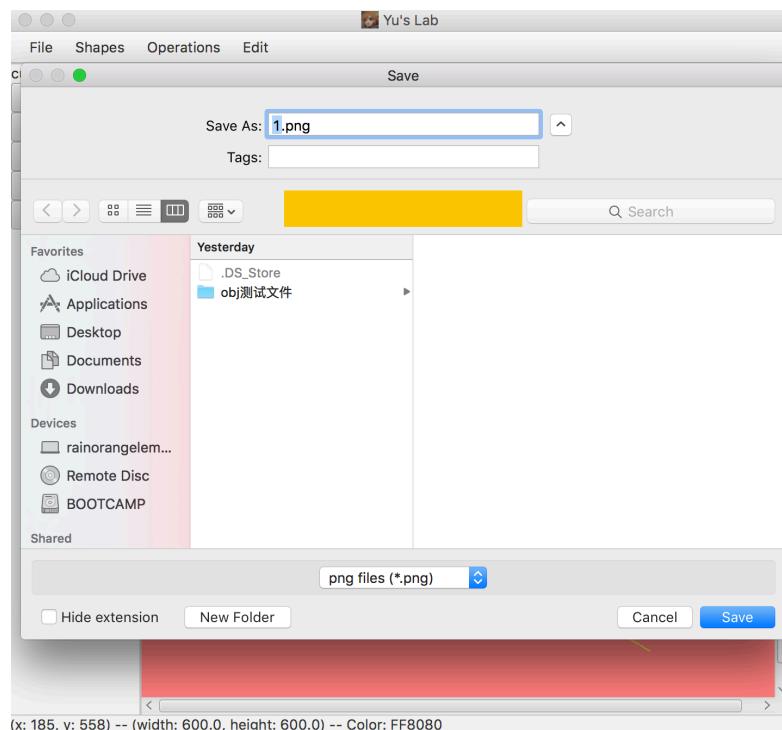


图 22: 导出图片仅支持 PNG 格式。可修改文件名与文件路径。需要注意，若画板中包含 3D 物体，则导出时会将包含 3D 物体的图像导出。若仅想导出画板上的 2D 图像，请通过 undo 删去导入 3D 图像的步骤，并添加导入 3D 后出现的 2D 的操作。

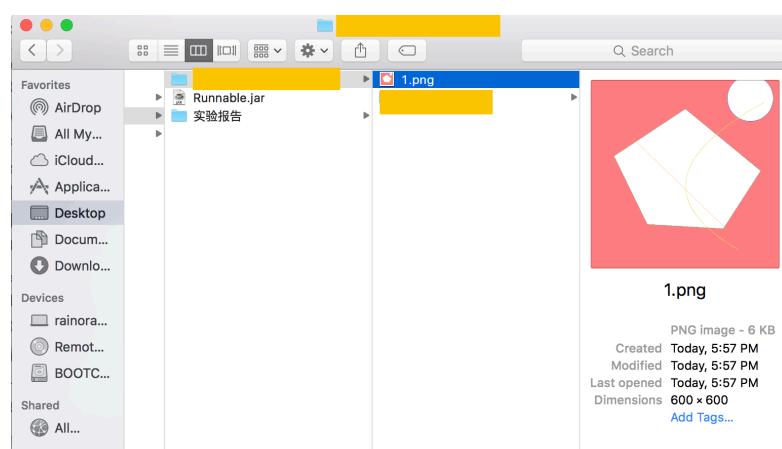


图 23: 成功导出。

3.3.2 导入图片

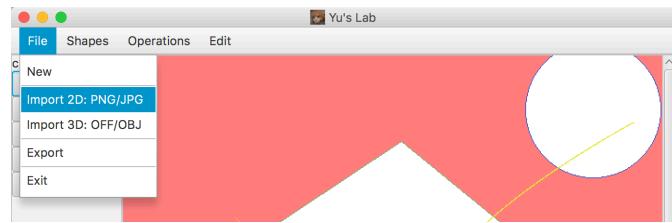


图 24: 在顶部菜单中选择: File→Import 2D: PNG/JPG

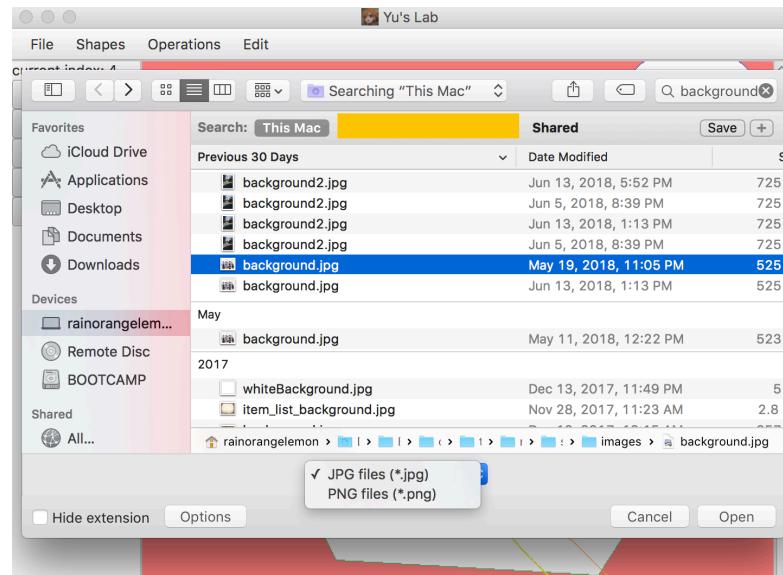


图 25: 导入图片支持 PNG 格式与 JPG 格式, 请自由选择。可选择文件路径。此处我们导入 JPG。

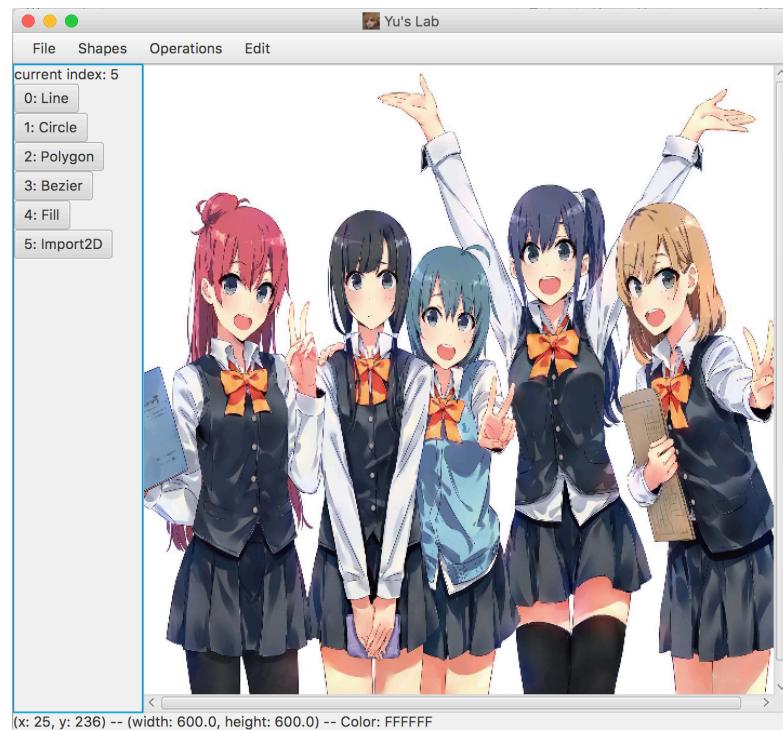


图 26: 成功导入! 需要注意, 图片会自动调整至画板大小, 并覆盖整个画板。若要调整图片大小, 使用缩放操作可以做到。若要将之前的图元覆盖在导入后的图片上, 请用户重新调整输入顺序, 先导入图片, 再输入图元。

3.4 3D 文件：导入

导入支持两种格式，OFF 格式与 OBJ 格式。接下来一一说明。由于 OBJ 格式相较于 OFF 格式，情况更加复杂，因此本节之后，我们仍以导入 OBJ 格式的情况进行说明。对于导入 OFF 格式的情况，读者可以参考导入 OBJ 格式的情况，很容易地进行推想。

3.4.1 3D 文件：导入 OFF 文件



图 27: 在顶部菜单中选择: File→Import 3D: OFF/OBJ

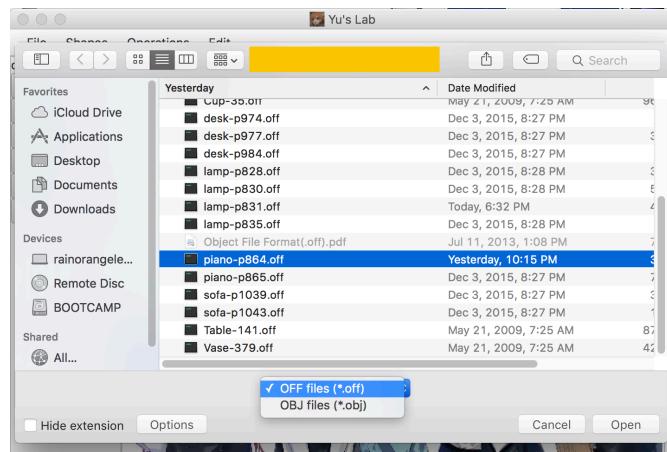


图 28: 此处我们导入给定的三维模型 piano-p864.off。在菜单下方，我们可以选择是导入 OFF 还是 OBJ。

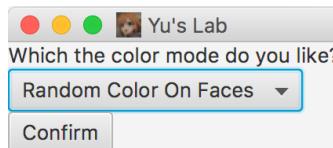


图 29: 由于 OFF 模型并没有给予关于填色的任何信息，因此系统会提供两套填色方案：用户自己决定物体表面的颜色；或者系统给每一个面随机填色。系统默认是随机填色，因为这种方案能够将面与面区分得较清楚。

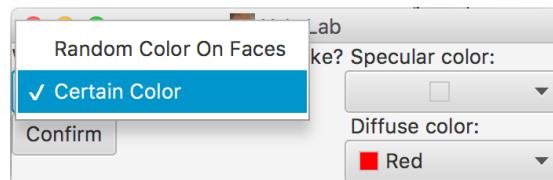


图 30: 另一种方案中，用户可以指定物体表面的漫反射色与镜面反射色。diffuse color 定义了在漫反射光照射下物体的颜色。specular color 设置的是物体受到的镜面光照的影响的颜色。

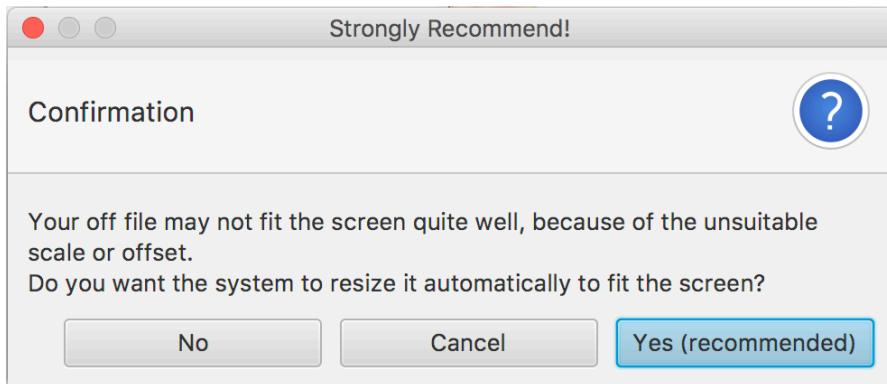


图 31: 在设置完颜色后, 系统会检查 OFF 文件中的物体是否不太符合屏幕的尺寸, 不符合的情况包括: 尺寸太小, 尺寸太大, 或只有物体的一小部分会显示在屏幕上, 或 z 轴坐标均大于 0。在这些情况下, 系统会跳出如上的弹窗, 建议用户允许其根据画板大小, 对于 Off 文件的顶点坐标进行调整, 以符合画板尺寸。为了美观, 建议用户选择 Yes, 否则可能因为顶点坐标范围太小, 导致物体在画板上被缩成了一个小点。

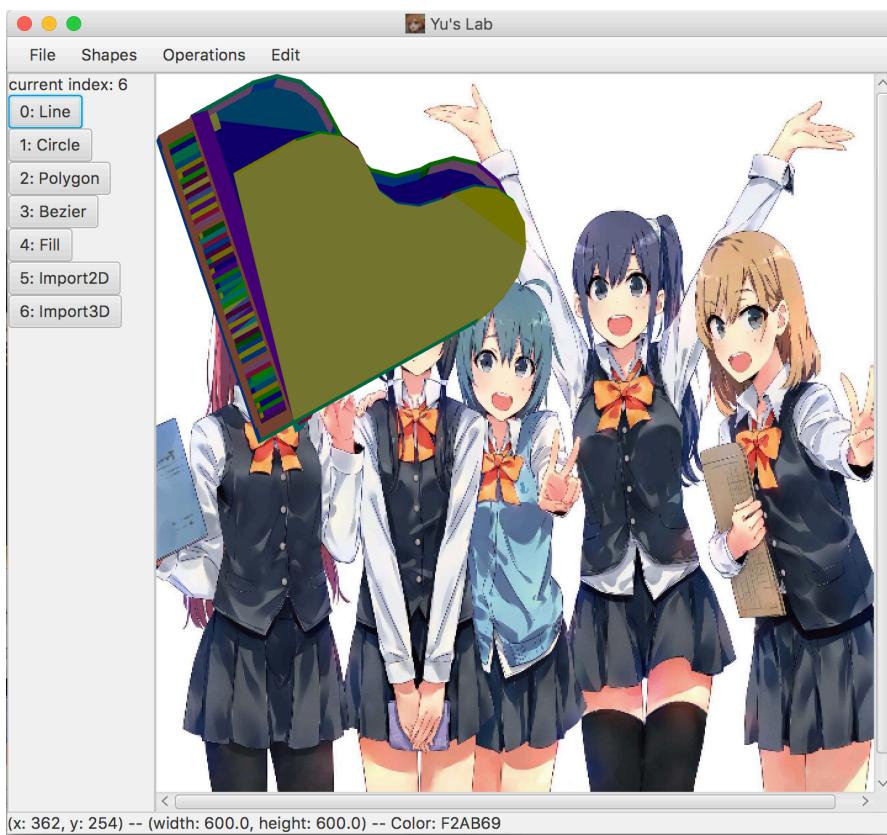


图 32: 成功导入一架钢琴! (可怜的绘麻和静香) 可以看到, 钢琴每一个面被随机涂色了。之后我们会以 OBJ 文件为例, 介绍三维模型的平移、旋转和缩放。OFF 文件是 OBJ 文件的更简单的形式, 因此也能够参照之后的说明进行操作。

3.4.2 3D 文件：导入 OBJ 文件



图 33: 在顶部菜单中选择: File→Import 3D: OFF/OBJ

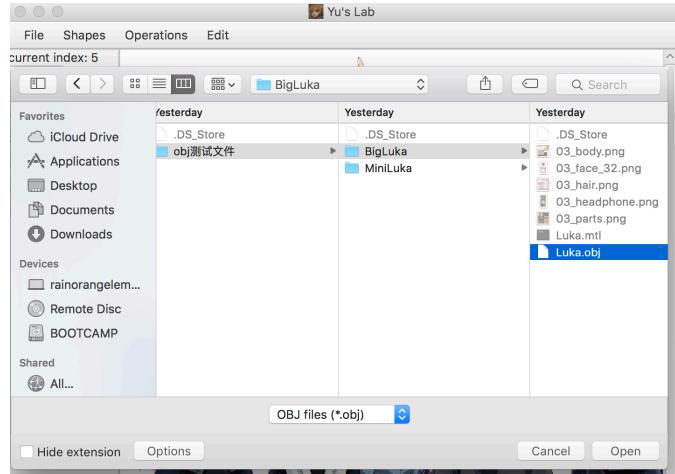


图 34: 在弹出窗口的底部选择导入的文件格式为 OBJ, 然后导入目标 OBJ 文件。此处我们导入 Resources 文件夹中包含的测试文件 Luka.obj。

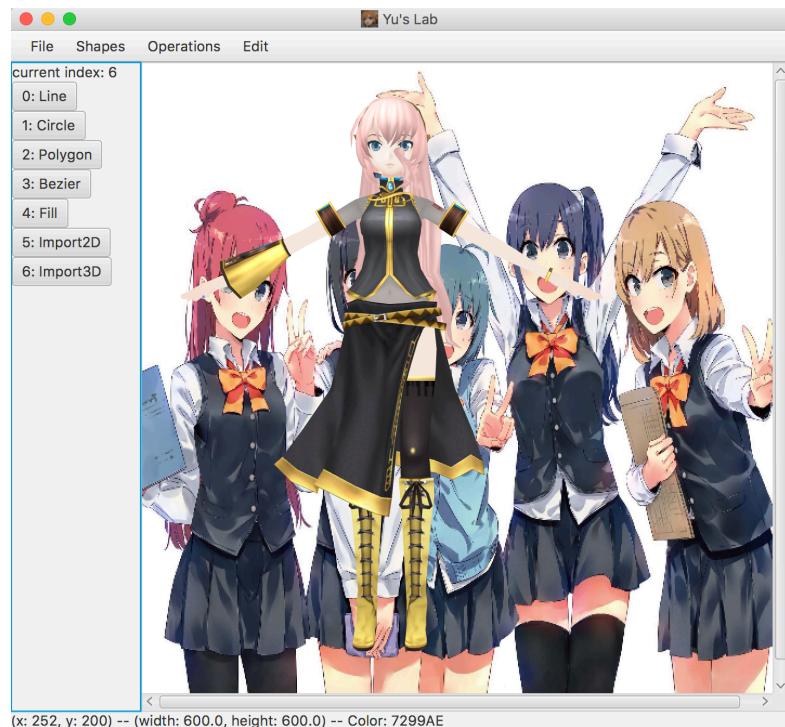


图 35: 成功导入! (可怜的绘麻 x 2)

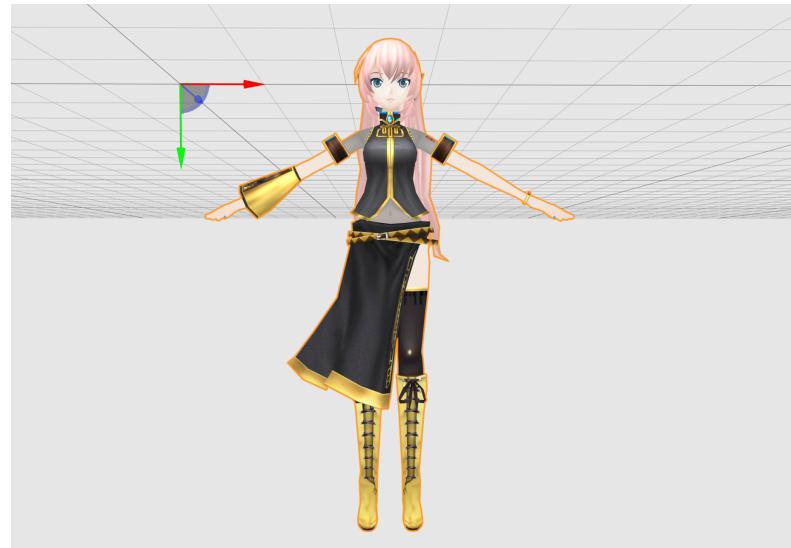


图 36: Luka 在画布中的位置, OFF 文件也能够参照上图: 如图所示, x 轴为红色, y 轴为绿色, z 轴为蓝色。画布所处 z 轴坐标为 0。因此需要注意, 3D 物体仅会显示 z 轴坐标小于零的部分。因此, 若某 3D 物体内所有点的 z 轴坐标均大于零, 则其会被画板所覆盖, 不被显示。此外, 如果由某 3D 物体在画面中仅显示了它的局部, 请尝试在原 off 或者 obj 文件或在系统中, 将物体往 z 轴反方向进行平移。

3.5 图形数据：裁剪

我们来尝试裁剪一下图片。



图 37: 在顶部菜单中选择: Operations→Clip

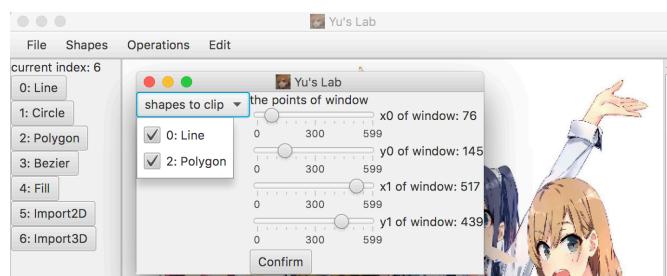


图 38: 选择指定的裁剪窗口的左上角与右下角。若 x_0 大于等于 x_1 , 系统会自动对调 x_0 与 x_1 。同理与 y_0 和 y_1 。shapes to clip 中包含之前生成的线与多边形。若其中某图元被选择，则其被裁剪后的图元会被重新生成。默认重新生成的线与多边形为全选。



图 39: 裁剪成功。由于在裁剪阶段，裁剪后的多边形与线被重新生成，且裁剪操作在导入图片操作之后，因此被重新生成的多边形和线会覆盖在导入的图片之上。3D 物体不支持裁剪。

3.6 图形数据：变换

以下我们对 3D 物体进行变换，二维情况会更加简单。针对不同的图元会有不同的变换行为，这在本节最后会进行列举。

3.6.1 变换：缩放

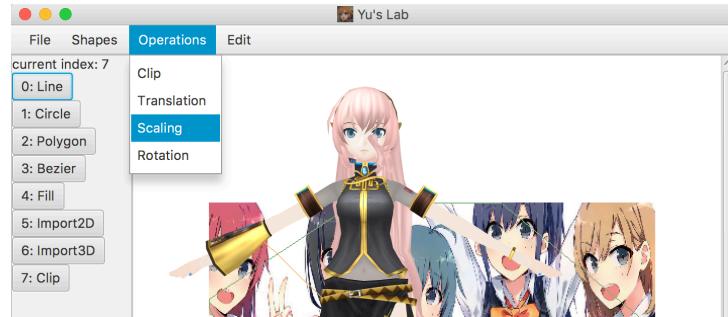


图 40: 在顶部菜单中选择: Operations→Scaling

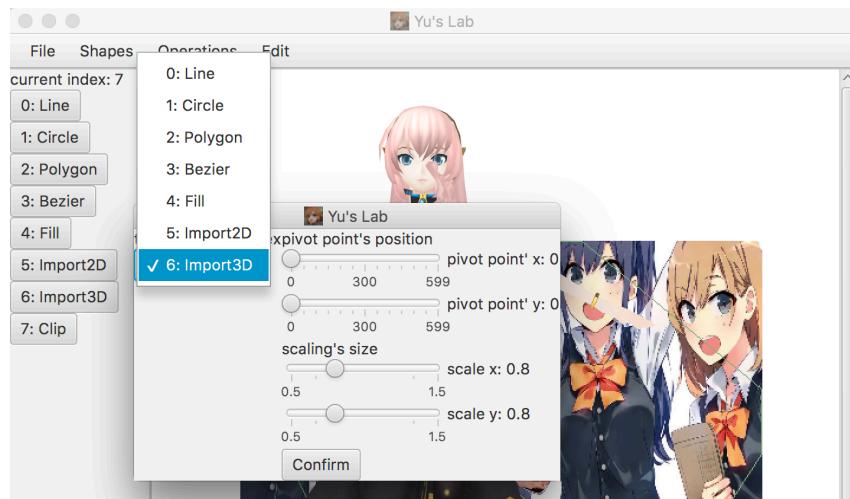


图 41: 选择缩放对象，缩放中心点的位置，以及缩放大小。

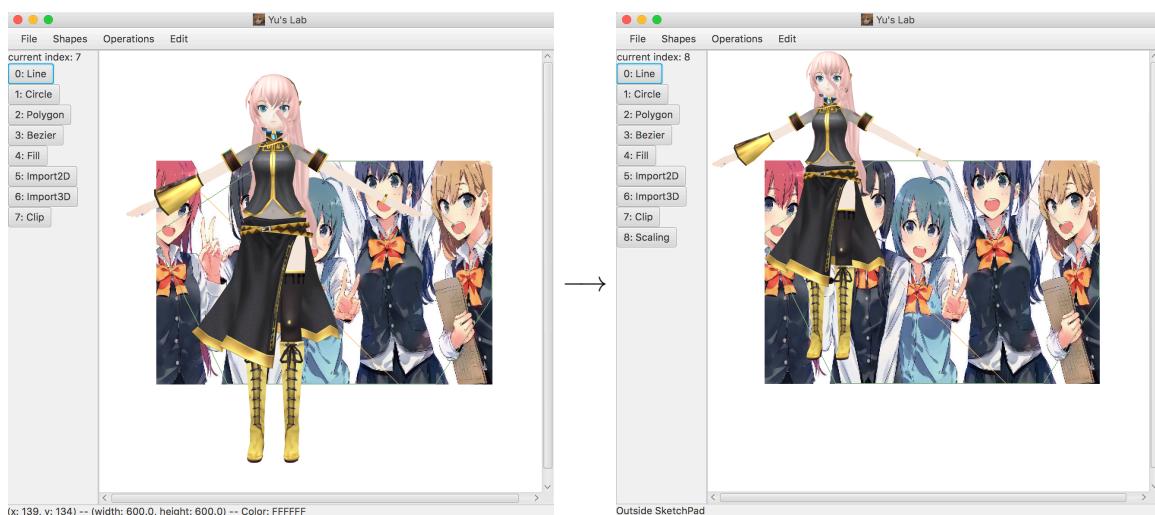


图 42: 缩放成功。

3.6.2 变换：旋转



图 43: 在顶部菜单中选择: Operations→Rotation

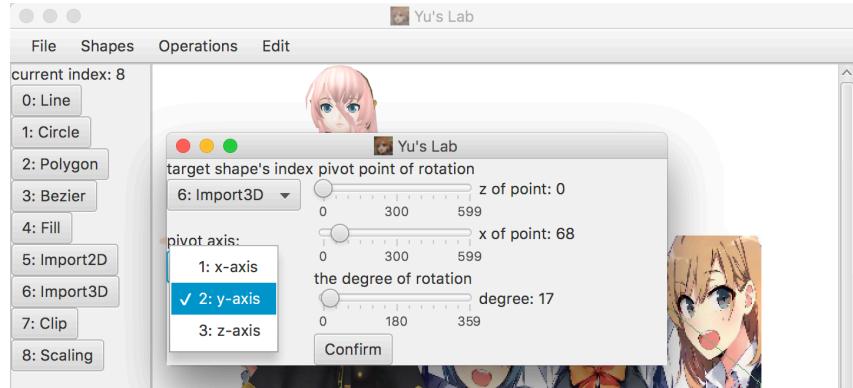


图 44: 选择旋转对象，旋转中心点的位置，以及旋转角度。特别地，针对 3D 物体可以选择三种旋转轴，对应于三个维度。

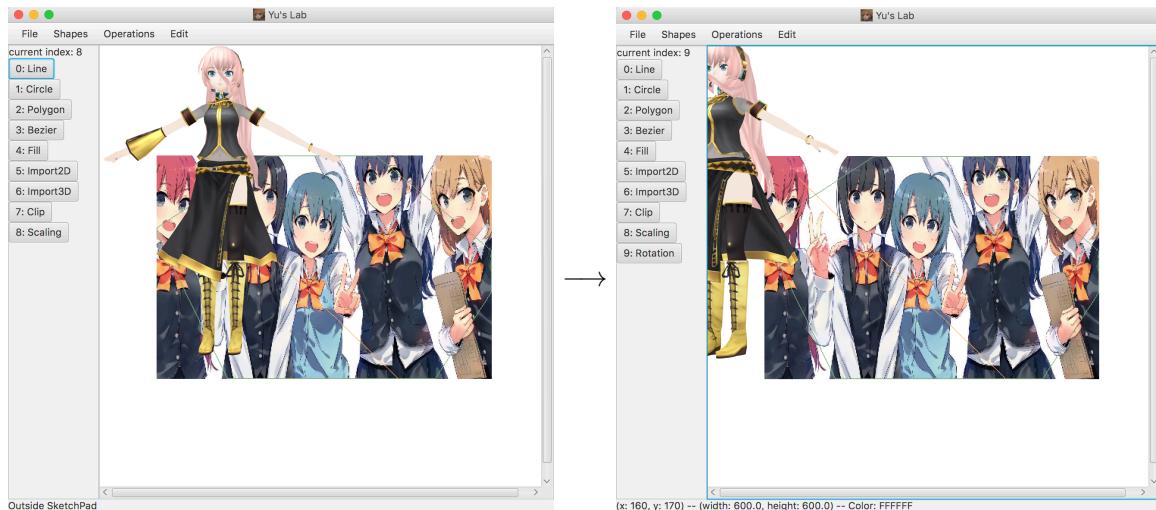


图 45: 旋转成功。

3.6.3 变换：平移

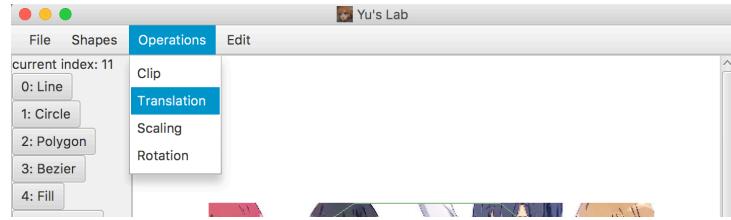


图 46: 在顶部菜单中选择: Operations→Translation

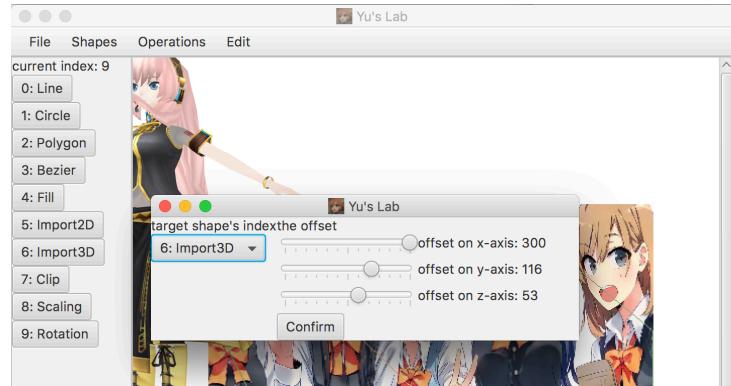


图 47: 选择平移对象，以及平移向量。特别地，针对 3D 物体，平移向量有三个维度；其余时候，平移向量为两个维度。

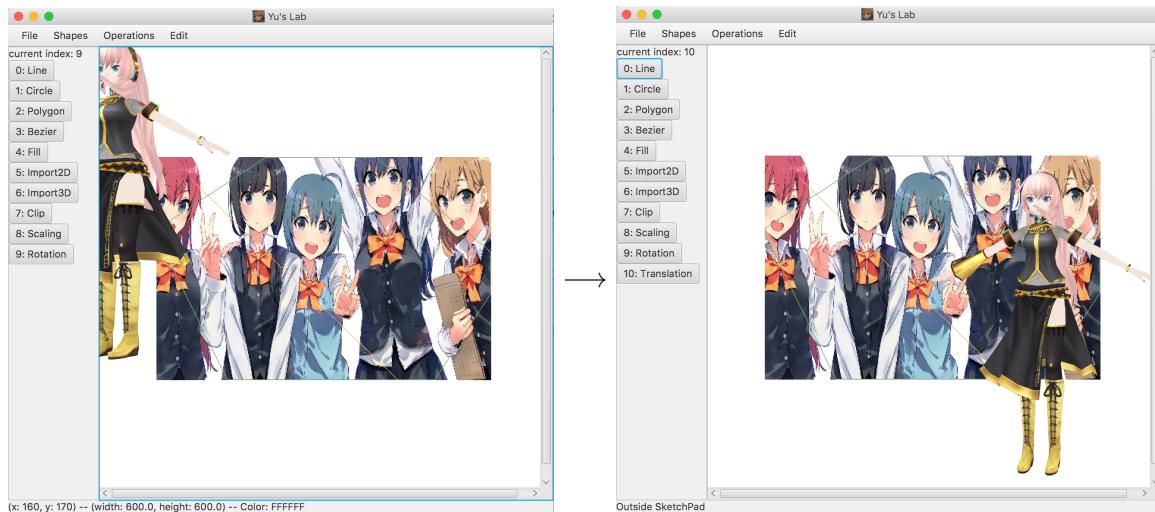


图 48: 平移成功。

3.6.4 变换：连续操作

由于以上操作是以滚动条形式输入数据的，而滚动条有上限和下限，带来了变换操作的局限。如在上面的平移操作中，平移向量的 x 轴分量上限为 300。因此，如果用户此时已将 Luka 平移 300 个像素，但仍想继续将 Luka 往右平移时，则需要对 Luka 进行再次变换操作。通过连续操作，我们也能实现对于三维物体的任意三维角度的旋转。

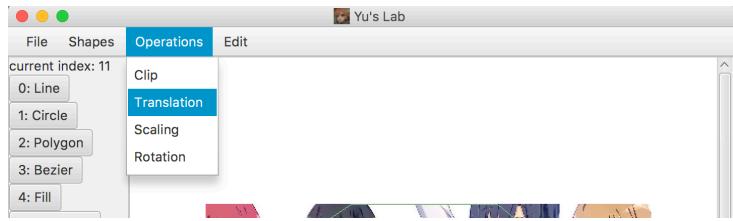


图 49: 在顶部菜单中选择再次变换操作的类型

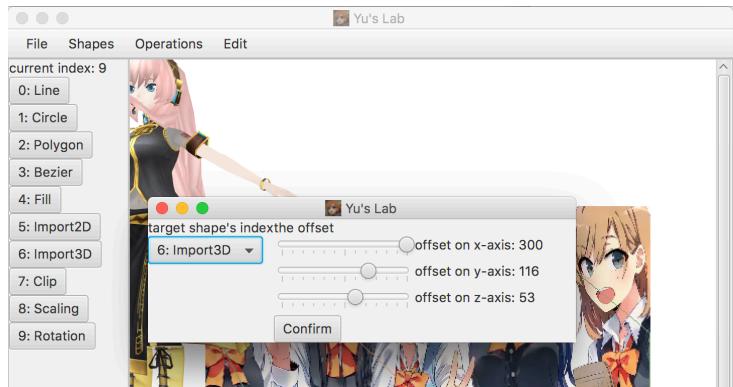


图 50: 输入再次变换的参数

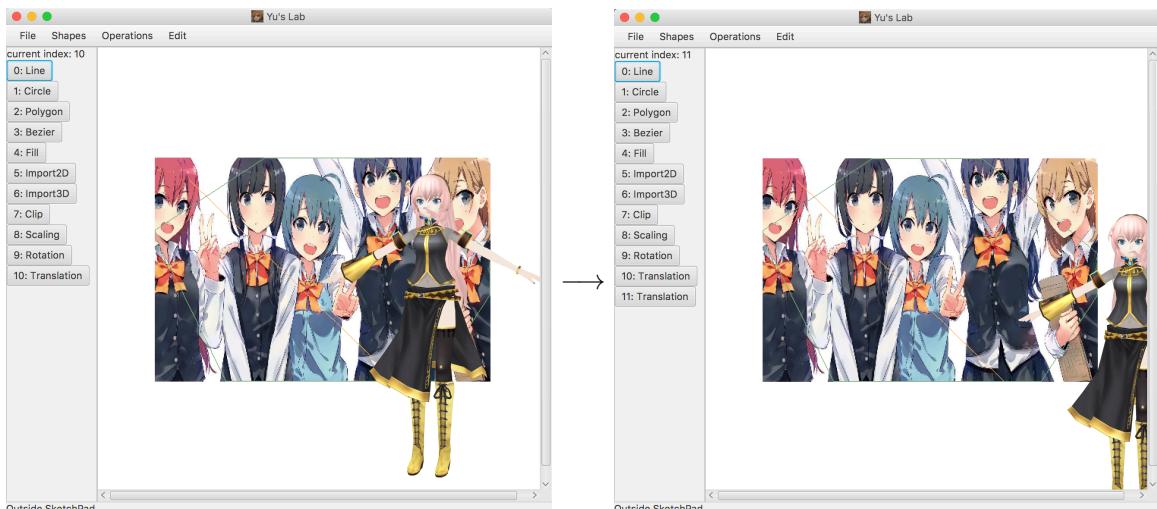


图 51: 连续变换成功。

3.6.5 变换：所支持不同图元的不同变换

图元类型	旋转	平移	缩放
直线/多边形/曲线/圆/椭圆	二维旋转	二维平移	二维缩放
填充	无变换	无变换	无变换
图片	二维旋转	二维平移	二维缩放
三维物体	三维旋转	三维平移	二维缩放 (x 轴坐标和 y 轴坐标可以进行缩放)

表 1: 不同图元的不同变换

3.7 图形 & 操作数据：编辑

在以上图中，我们可以看到在系统的最左侧有一系列的按钮，每个按钮对应了之前的一次操作：生成图元/导入图片/导入3D物体/变换操作/裁剪操作。我们可以通过点击这些按钮进行对于先前图元/操作的编辑，编辑完成之后画板会从头开始按操作的顺序进行刷新。例如：

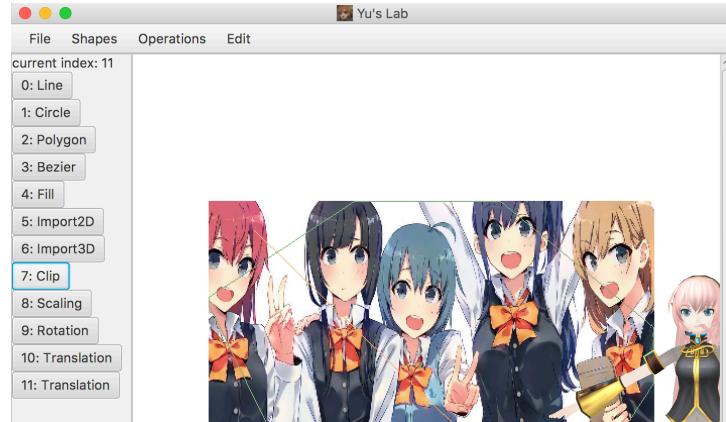


图 52：在左边按钮中选择想要修改的图元或者操作，此处我们选择 7: Clip

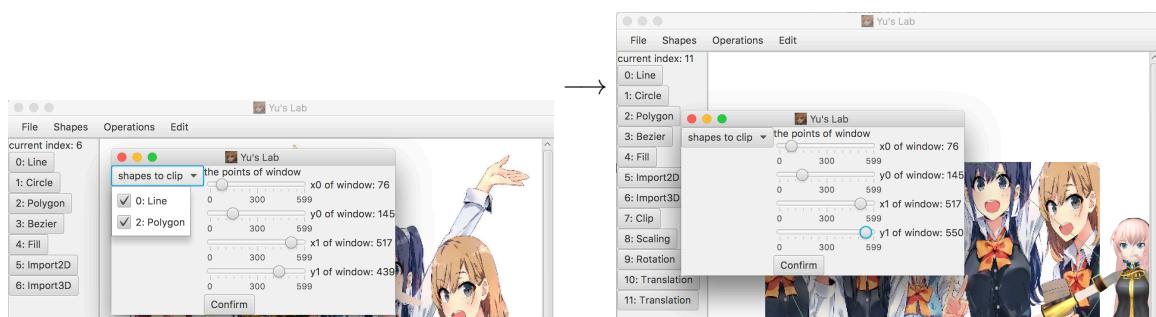


图 53：修改操作的参数或者图元的参数。此处我们向下拉长裁剪窗口。

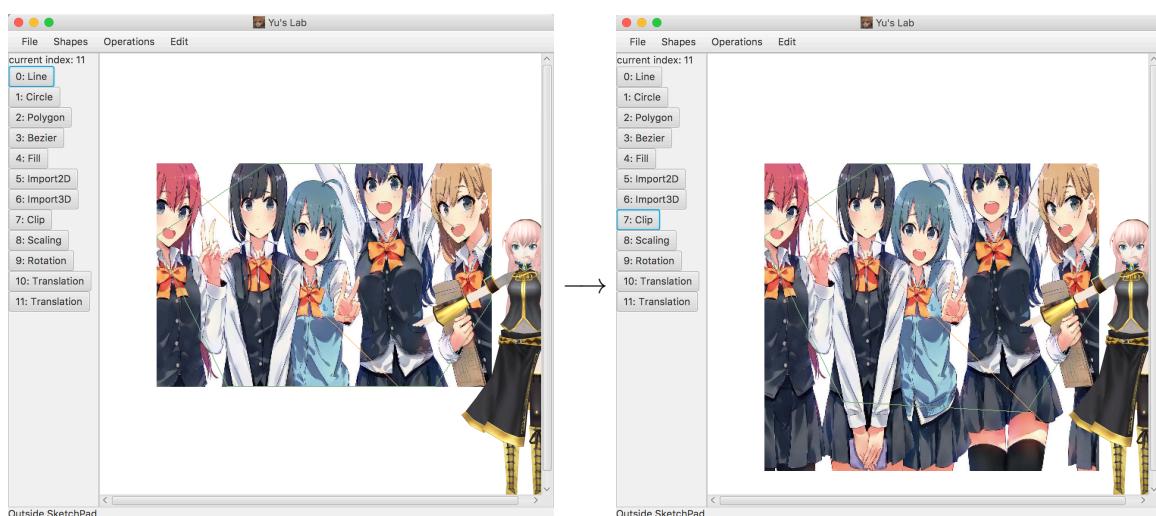


图 54：修改成功。需要注意，之前修改的物体可能经过之后用户定义的操作，达到与用户预想中未经过之后操作的物体不一样的结果。此时建议用户先回撤至物体刚生成的画板状态，再进行修改，修改完成后，再恢复至最新的画板状态。

3.8 撤销与恢复

3.8.1 撤销

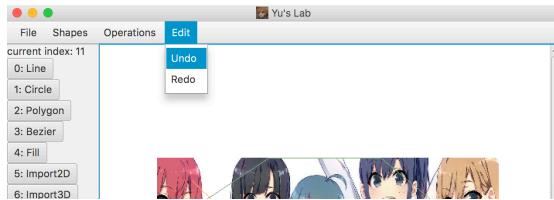


图 55: 在顶部菜单中选择: Edit→Undo

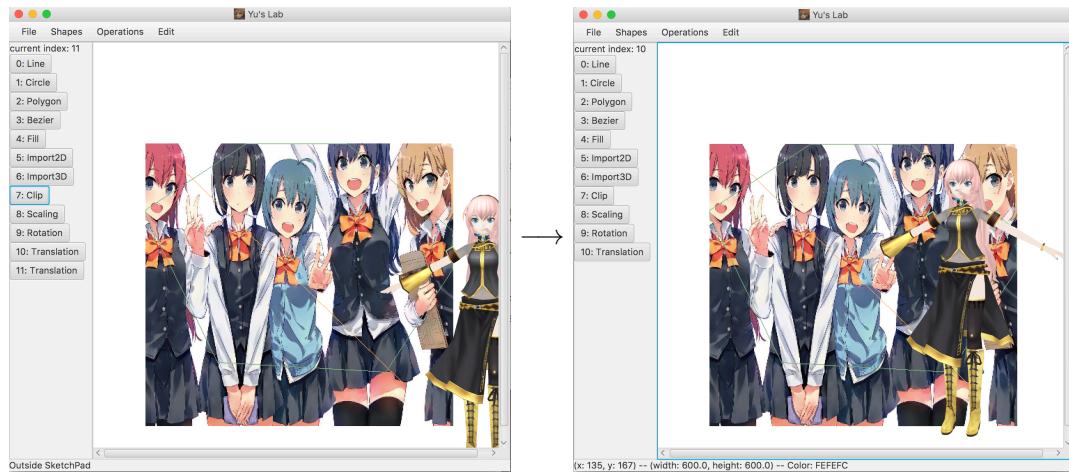


图 56: 修改成功。撤销掉最后一步 Translation, Luka 回到了相较于最后偏左的状态。

3.8.2 恢复



图 57: 在顶部菜单中选择: Edit→Redo

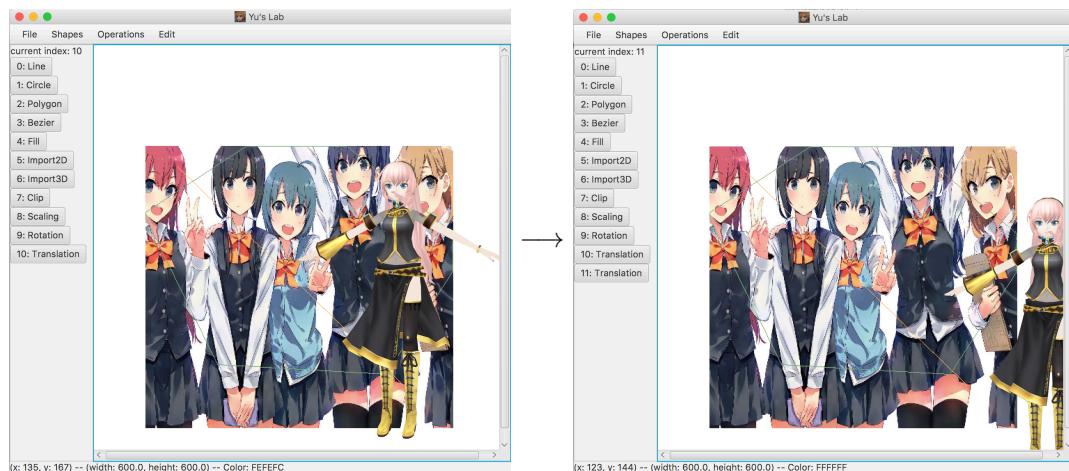


图 58: 修改成功。恢复了最后一步 Translation, Luka 回到了相较于倒数第二步偏右的状态。

4 系统介绍

4.1 目录树与模块代码介绍

接下来，对系统每个模块的代码和实现功能，以目录树的形式进行介绍：

```
src
├── Main.java (程序入口)
├── controller (C in MVC, 用于响应用户操作和调用模型 API)
│   ├── StartMenu.java (初始界面, 选择画板大小)
│   └── Board.java (整个画板界面, 包含菜单、各种按钮、和画布)
└── model (M in MVC, 用于封装模型操作)
    ├── ModelStep.java (操作步, ModelShape 与 ModelOperation 的父类)
    ├── ShapeManager.java (操作步数组, 封装 undo、redo、refresh)
    ├── ModelShape.java (抽象图元类, 有对应于生成图元、平移、旋转、缩放的抽象方法, 以下均为其继承)
    │   ├── ModelDot.java (点)
    │   ├── ModelDots.java (点阵)
    │   ├── ModelLine.java (线)
    │   ├── ModelCircle.java (圆/椭圆)
    │   ├── ModelPolygon.java (多边形)
    │   ├── ModelBezier.java (Bezier 曲线)
    │   ├── ModelFill.java (填充)
    │   ├── ModelImport2D.java (导入 PNG 或 JPG 图片)
    │   └── ModelImport2D.java (导入 OFF 或 OBJ 模型)
    ├── model.operation. (包含对于图元的操作) .3 ModelOperation.java (对图元的操作, 有抽象方法 operate, 以下均为其继承)
    │   ├── OperationClip.java (裁剪)
    │   ├── OperationRotation.java (旋转)
    │   ├── OperationScaling.java (缩放)
    │   └── OperationTranslation.java (平移)
    └── view (V in MVC, 用于显示画板和 UI)
        ├── Dot.java (封装画板上的点)
        └── SketchPad.java (画板)
ui.chooser
    ├── ShapeChooser.java (图元的编辑 UI, 有抽象方法 showEditor, 以下均为其继承)
    ├── LineChooser.java (线段的编辑 UI)
    ├── PolygonChooser.java (多边形的编辑 UI)
    ├── BezierChooser.java (Bezier 曲线的编辑 UI)
    └── CircleChooser.java (圆/椭圆的编辑 UI)
```

```
    └── FillChooser.java (填充算法的编辑 UI)
    └── ImportChooser.java (导入二维模型的编辑 UI)
    └── ImportChooser.java (导入三维模型的编辑 UI)

ui.operator
    └── Operator.java (操作编辑 UI, 有抽象方法 showEditor, 以下均为其继承)
        └── ClipOperator.java (裁剪编辑 UI)
        └── RotationOperator.java (旋转编辑 UI)
        └── ScalingOperator.java (缩放编辑 UI)
        └── TranslationOperator.java (平移编辑 UI)

util (工具包, 包含一些与外部资源文件交互的方法, 以及常用的静态方法)
    └── UIComponentFactory.java (封装生成常用的 UI 部件方法)
    └── File3dImporter (抽象类, 包含读入 3d 文件, 以及生成三角网格的抽象方法)
        └── OFFImporter (继承 File3dImporter, 用于读入 OFF 文件, 并生成三维图元)
        └── ObjImporter (继承 File3dImporter, 用于读入 OBJ 文件, 并生成三维图元)
        └── MtlReader (用于读入 Mtl 文件, 生成渲染的材质 (PhongMaterial), 导入 OBJ
            文件时被使用)
        └── SmoothingGroup (光滑组, 用于使一系列面变为平滑的曲面, 在导入 OBJ 文件时
            被使用)
    └── MeshStart (用于记录每个面在各数组中的起始位置与终点位置, 在导入 OBJ 或
        OFF 文件时被使用)
```

4.2 关于 License 的说明

在观察类 ObjImporter, MtlReader, SmoothingGroups 的时候, 我们可以在代码顶部看见如下的注释:

「此函数由 Rainorangelemon 在原代码基础上进行重构与更改, 已完全理解原代码所含内容。」

Copyright (c) 2010, 2014, Oracle and/or its affiliates.

All rights reserved. Use is subject to license terms ...

」

我需要说明, 我这几部分的源代码借鉴自 openjfx 团队的 3DViewer, 并已经过我的重(魔)构(改), 其能够与系统其他模块深度交互, 一起健壮地工作, 就是我重构成功最好的证明。我也曾犹豫是否要把原注释删去, 毕竟我重构了许多代码。但为了尊重开发者的劳动智慧, 我还是贴了上去。因此, 如果有人需要借鉴这些包含注释代码的话, 也最好请保留这段注释。对于其余的代码部分, 请遵守 Apache License 2.0 协议。

5 总结

通过自己实现这个系统，我掌握了许许多多图形学的知识与算法：直线与圆的生成算法，矢量图与像素图，多边形凸检验、直线与多边形裁剪算法、二维变换算法、Bezier 曲线、椭圆生成算法 …；同时，我也对如何有机组合这些图形操作有了自己的认知。通过 Java 面向对象式编程思想的帮助，我成功地花费无数个日日夜夜，实现了一个健壮的、自洽的图形操作软件。我记得自己第一次导入 OFF 格式，却发现画板一片空白的失落；也仍能回想起填充算法和裁剪算法产生冲突时，自己的焦虑；最不能忘怀的，是第一次在茫茫一片白色的画板上，显示出尚不能调整颜色的直线时，自己的喜悦。感谢图形学课程提供给我的这一次机会，让我不断在 bug 中挑战自我，锻炼了我的 code 能力，并一遍遍地完善自己。

希望用户小伙伴们使用愉快，画出漂亮的图片！