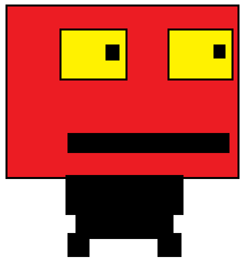
Course Name: CSE 300
Student Name: Yunlong Zhang
Student ID: 107407111



ANIMATED SPRITE EDITOR 1.0 USER'S MANUAL

These users guide was written for the application users to familiarize themselves with **Animated Sprite Editor**. These instructions were written for public, and user does not need any knowledge of this program. The user should have fundamental computer knowledge and know the basis of how to operate Java on NetBean environment.

What Is Animated Sprite Editor?

Animated Poseur Editor is an educational project that was developed under Professor McKenna's class Computer Science III in Stony Brook by Author Yunlong Zhang. The majority of the audience consists of nonspecialists and experts. The current version of **Animated Sprite Editor** is 1.0. Further update will be available with bug-detection, new features plug-in and software maintenance.

Animated Poseur Editor is open source software to help users design a 2D game character along with imagination and creativity. The sprite can be customized in a form of animations state. Each animation state includes many chained poses. For each individual pose, the painting canvas enable user to draw images and store images in a particular file format named pose. XML format file is used in this project to store all data including sprite information, animation state and pose.

Animated Poseur Editor is a fun tool for people to use. This project is aiming to help user construct animated sprites for the purposes of entertainment and education.

Animated Sprite Editor - Setting Up

What equipment you will need:

- A computer with fundamental functions installed, and graphics drawing tablet mouse pad and a high precision DPI mouse recommended.
- NetBean IDE 7.2 or7.3 is a powerful integrated development environment application on the Java platform. The NetBean IDE 7.2 software takes up about at least 300 MB of disk for running **Animated Sprite Editor**.
- Java JDK 7 takes up about 130MB disk space for Windows 64 bits system.
- The **Animated Sprite Editor** project takes up about 8MB disk space.

Installation:

To install the NetBean IDE and JDK 7, perform the following steps:

1. Start Internet Browser. (IE, Chrome, Mozilla, etc.)

2. Browse to the following URL

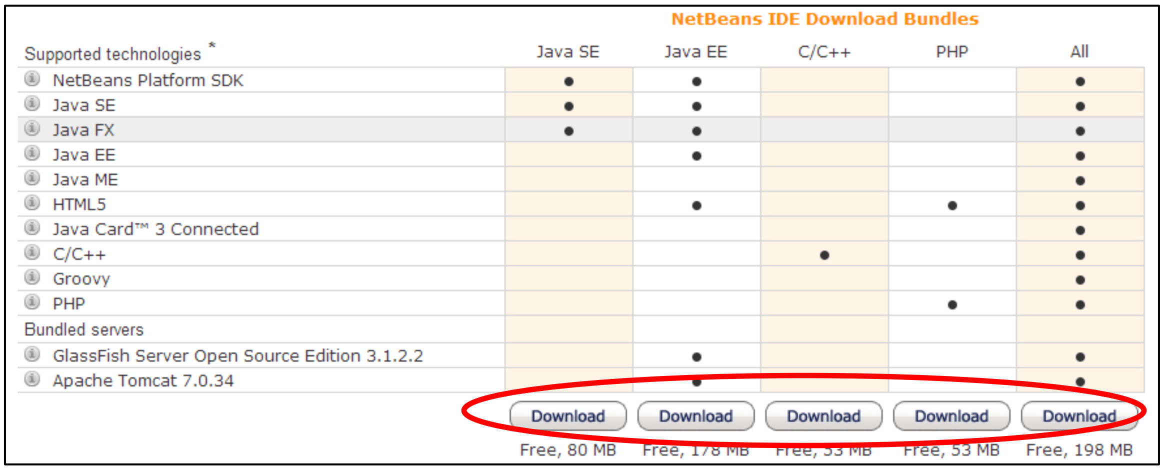
   `http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html` (JDK 7)

   `http://netbeans.org/downloads/` (NetBean IDE)

   The `Download` Menu appears.

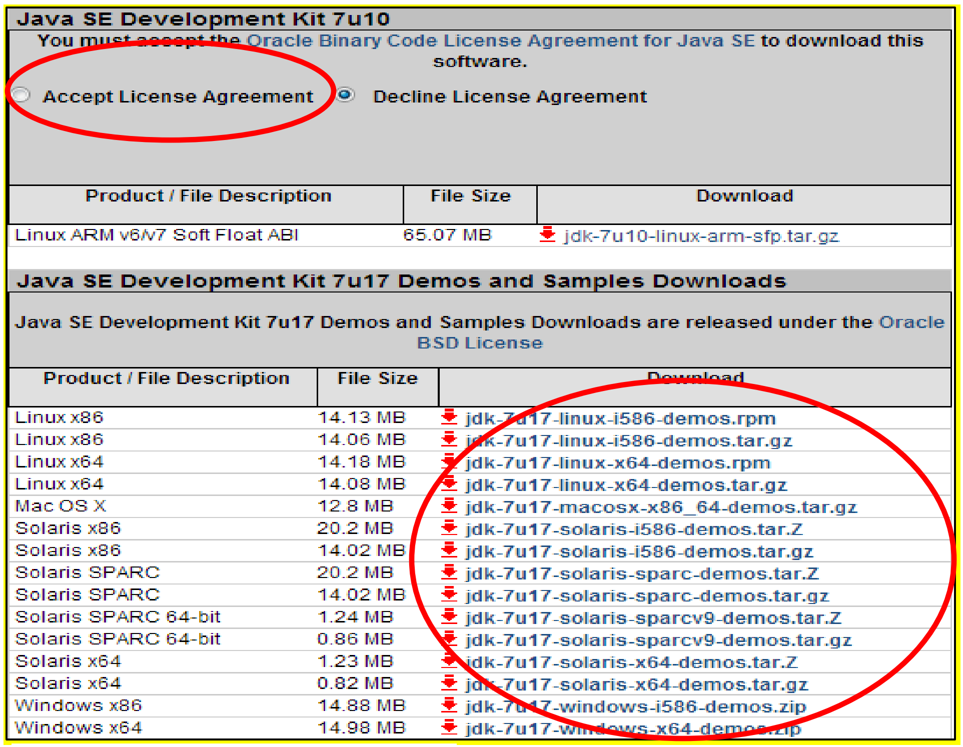
3.

   The main download pages look like:



	NetBeans IDE Download Bundles				
Supported technologies *	Java SE	Java EE	C/C++	PHP	All
NetBeans Platform SDK	●	●			●
Java SE	●	●			●
Java FX	●	●			●
Java EE		●			●
Java ME					●
HTML5		●		●	●
Java Card™ 3 Connected					●
C/C++			●		●
Groovy					●
PHP				●	●
Bundled servers					
GlassFish Server Open Source Edition 3.1.2.2		●			●
Apache Tomcat 7.0.34		●			●
	Download	Download	Download	Download	Download
	Free, 80 MB	Free, 178 MB	Free, 53 MB	Free, 53 MB	Free, 198 MB

Figure 1, Downloading NetBean IDE site



Java SE Development Kit 7u10

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement [✓ Decline License Agreement]

Product / File Description	File Size	Download
Linux ARM v6/v7 Soft Float ABI	65.07 MB	jdk-7u10-linux-arm-sfp.tar.gz

Java SE Development Kit 7u17 Demos and Samples Downloads

Java SE Development Kit 7u17 Demos and Samples Downloads are released under the Oracle BSD License

Product / File Description	File Size	Download
Linux x86	14.13 MB	jdk-7u17-linux-i586-demos.rpm
Linux x86	14.06 MB	jdk-7u17-linux-i586-demos.tar.gz
Linux x64	14.18 MB	jdk-7u17-linux-x64-demos.rpm
Linux x64	14.08 MB	jdk-7u17-linux-x64-demos.tar.gz
Mac OS X	12.8 MB	jdk-7u17-macosx-x86_64-demos.tar.gz
Solaris x86	20.2 MB	jdk-7u17-solaris-i586-demos.tar.Z
Solaris x86	14.02 MB	jdk-7u17-solaris-i586-demos.tar.gz
Solaris SPARC	20.2 MB	jdk-7u17-solaris-sparc-demos.tar.Z
Solaris SPARC	14.02 MB	jdk-7u17-solaris-sparc-demos.tar.gz
Solaris SPARC 64-bit	1.24 MB	jdk-7u17-solaris-sparcv9-demos.tar.Z
Solaris SPARC 64-bit	0.86 MB	jdk-7u17-solaris-sparcv9-demos.tar.gz
Solaris x64	1.23 MB	jdk-7u17-solaris-x64-demos.tar.Z
Solaris x64	0.82 MB	jdk-7u17-solaris-x64-demos.tar.gz
Windows x86	14.88 MB	jdk-7u17-windows-i586-demos.zip
Windows x64	14.98 MB	jdk-7u17-windows-x64-demos.zip

Figure 2, Downloading JDK 7 site

- In the `Download` menu, *click* the `Download` and `Save` menu option regards to operating system for each individual download.
- When download NetBean, users *must* choose the supported technologies including **Java SE**, do *not* download other bundles without **Java SE**.
- Make sure *click* the **Accept License Agreement** before downloading JDK 7.

Important Note - Downloading time is related to the Internet speed. If a network failure has doomed the download, then wait a few minutes start over at step 1.

4. Execute the installation (Window 64 bits)

    - `../download/netbeans-7.3-javase-windows.exe`
    - `../download/dk-7u17-windows-x64.exe`

    Install those two packages that under the directory above on the computer.

5. Run NetBean on desktop

    Now users are able to open NetBean at start page. Next section will show the instruction how to open our project step by step to users.

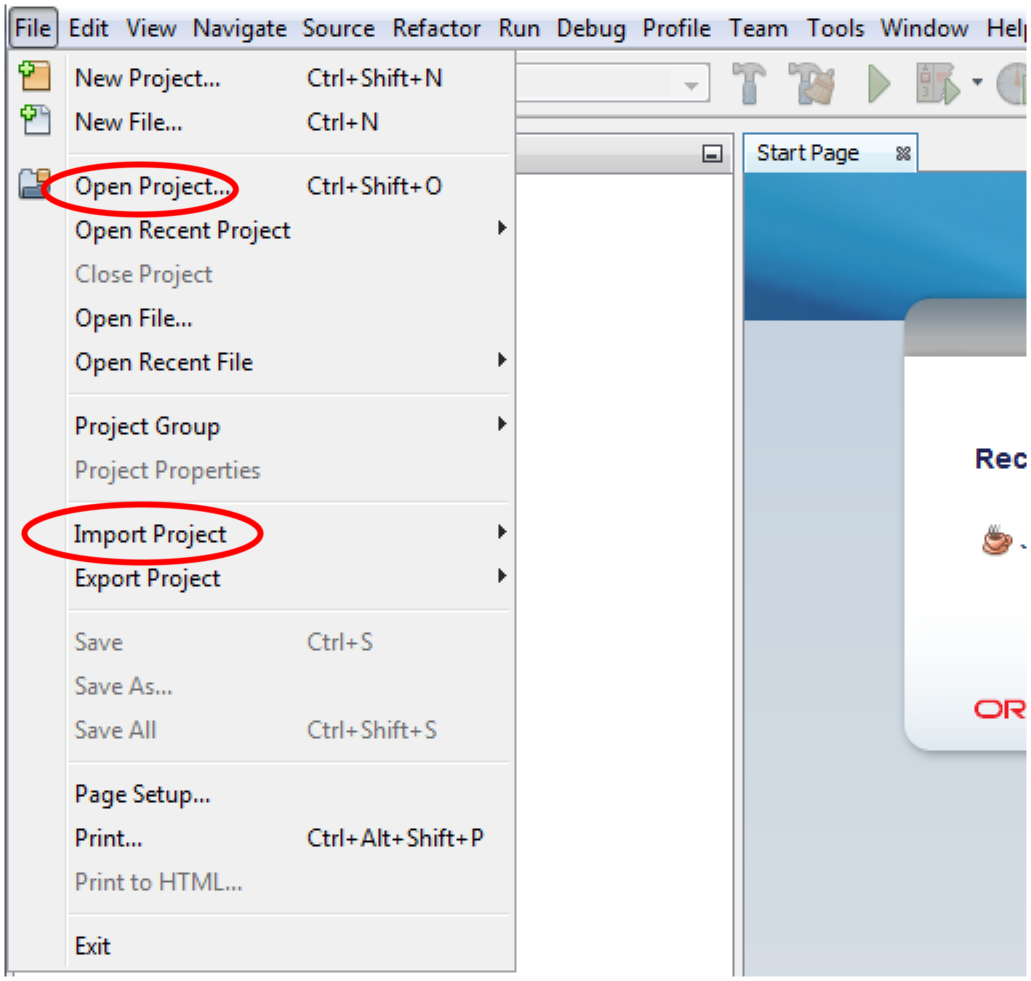


Figure 3, Starting page of NetBean

** In the other operating system, like Linux, Mac OS, installation process will be slightly different to Windows version. For instance, OS installation file will have* `'.dmg'` *file extension rather than* `'.exe'`*. However, the overall goal of this installation is identical in order to run the Animated Sprite Viewer in Java platform.*

Run Animated Sprite Editor on NetBean IDE

To open **Animated Sprite Editor**, perform the following steps:

1. Insert Flash Drive into USB port or CD into CD-ROM drive that carries project.

2. Copy the **Animated Sprite Editor** file to your hard drive.

   Important Note – Do *not* unzip the downloaded file. The original format of project file is `'.zip'`. Make sure that the full project file name is `AnimatedSpriteEditor.zip.`

3. Click Import Project and import `AnimatedSpriteEditor.zip` into NetBean as described in Figure 3.

4. Click Open Project and from now on there exists `AnimatedSpriteEditor` folder in projects list. Double click `AnimatedSpriteEditor` to open it.

5. In the project list located on left hand side of NetBean window, select `AnimatedSpriteEditor` in order to run the project, user can click the **Run Project** from Dropdown menu named **Run** at top, click green arrow button at buttons group, or simply click *F6* from keyboard.

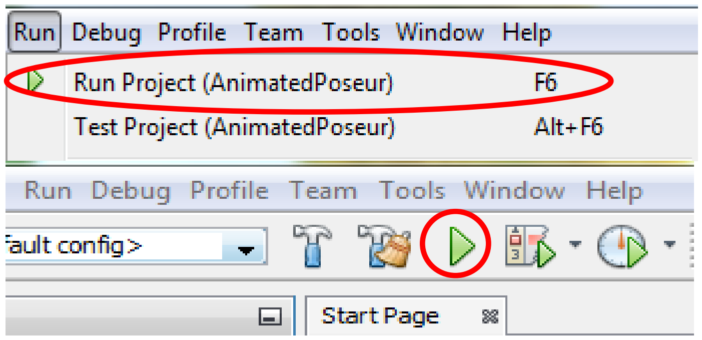
6. 
Run Debug Profile Team Tools Window Help
Run Project (AnimatedPoseur) F6
Test Project (AnimatedPoseur) Alt+F6
Run Debug Profile Team Tools Window Help
ault config>
Start Page

Figure 4, methods to run the project

7. After successfully launch project, users are able to see the start page as described in Figure 5.

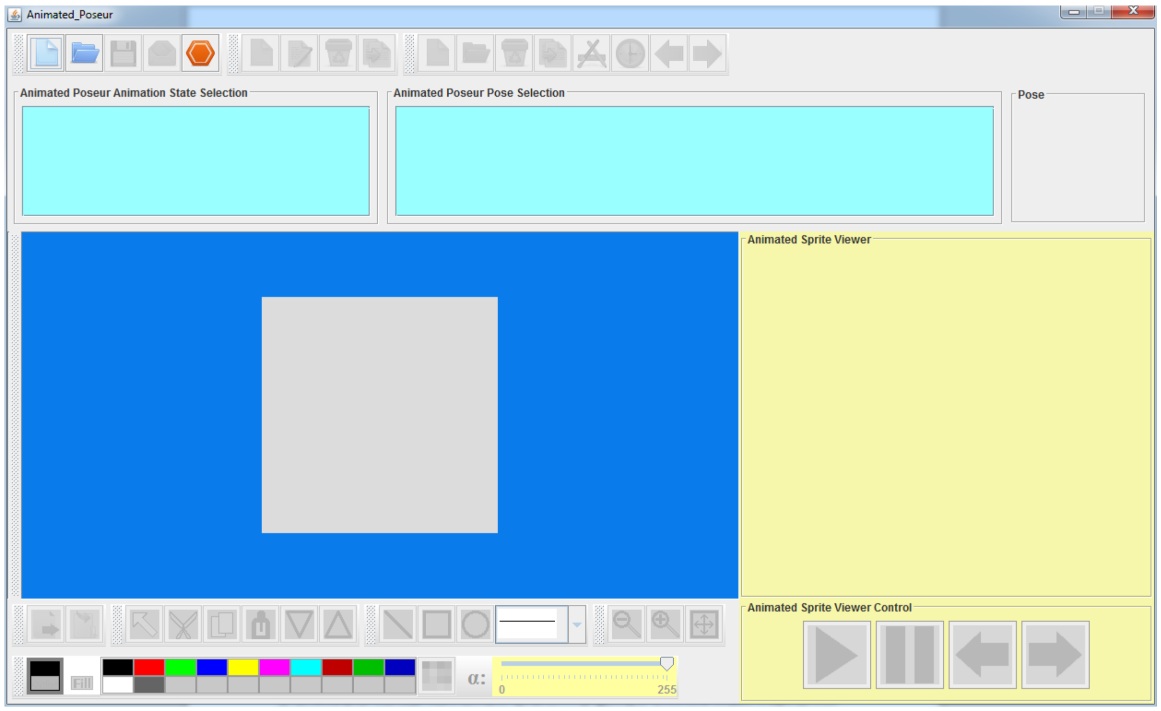

Animated_Poseur
Animated Poseur Animation State Selection
Animated Poseur Pose Selection
Pose
Animated Sprite Viewer
Animated Sprite Viewer Control
Fill
α:
0
255

Figure 5, Animated Sprite Editor Interface

What Makes Up a Sprite?

Initial project will not contain any sprite data unless users have used and created sprites before. In Animated Sprite Editor, users can not access and modify more than one sprite at same time. Each modification can only be applied to a single sprite. Furthermore, project contains three main properties:

- **Animated Sprite** – each sprite is made up by many animation states.
- **Animation State** – each sprite can have many animation states.
- **Animation State Pose** – each animation state was made up by many animation state poses.

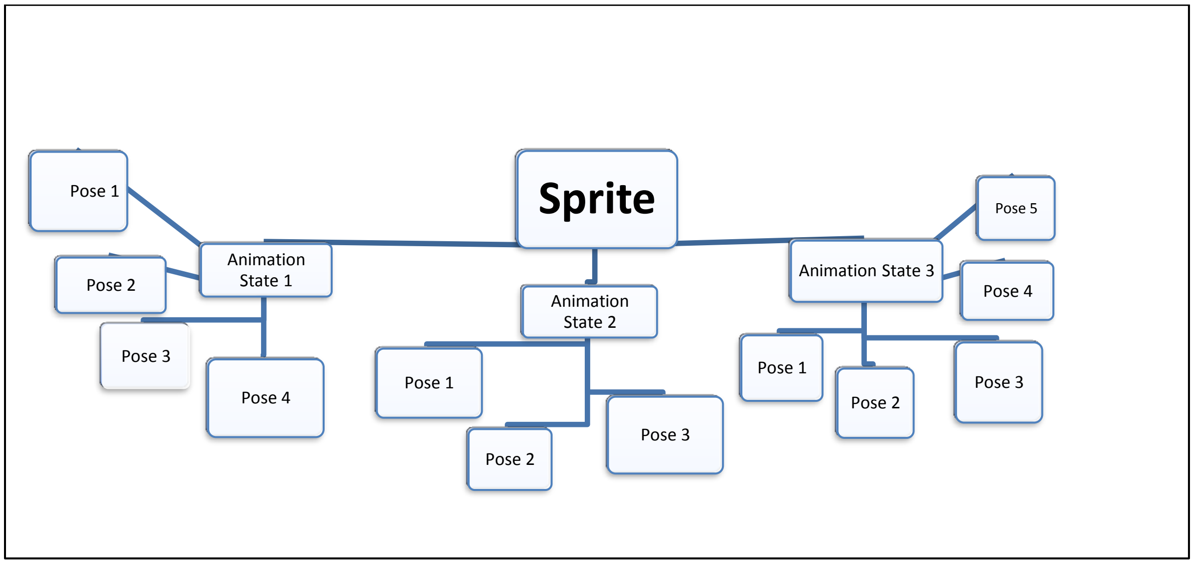


Figure 6, tree construction map of a sprite

Mostly `Hashmap` is widely used in this project to store data like **Animation State** or **Animation State Poses**. Then the customized data structure called Animated Sprite is the object that stores instance variables `Hashmap` and other data structures like `LinkedList` or `Array`. Figure 6 describes the relationship between each property.

On left sub-panel of north panel, there is a buttons group which is responsible of **Animated Sprite** Control, such as:

- `New Animated Sprite` button
- `Open Animated Sprite` button
- `Save Animated Sprite` button
- `SaveAs Animated Sprite` button
- `Exit Application` button

On center panel of north panel, there is a buttons group which is responsible of **Animation State** Control, such as:

- `New Animation State` button

- `Rename Animation State` button
- `Delete Animation State` button
- `Duplicate Animation State` button

On right panel of north panel, there is a button group which is responsible of **Animation State Pose** Control, such as:

- `New Pose` button
- `Open Pose` button
- `Delete Pose` button
- `Duplicate` Pose button
- `Edit Pose` button
- `Set Duration` button
- `Move Left` button
- `Move Right` button

Create an Animated Sprite

1. Click the `New Animated Sprite` button in **Animated Sprite** Control.

2. Enter your *Sprite Name* and click **OK.**

   Important Note - Users can *not* create a sprite with a name existed in the current data file. *No* empty input allowed in input dialog. Symbols and numbers are acceptable characters.

3. Notification dialog with message "`Poseur File has been saved`" popped up if sprite created.

4. Finish creating sprite and users can be able to edit sprite from then on.

   Important Note – all other buttons such as `Save Animated Sprite`, `SaveAs Animated Sprite` will be enabled after sprite created. Those buttons are helpful when users frequently save their art work in case of loss of sprite data.

Create an Animation State inside an Animated Sprite

1. Click the `New Animation State` button in **Animation State** Control.

2. Enter your *Animation State Name* and click **OK.**

   Important Note - Users can *not* create an `Animation State` with a name existed in the current data file. *No* empty input allowed in input dialog. Symbols and numbers are acceptable characters.

3. Finish creating animation state and users can be able to edit poses from then on.

   Important Note – all other buttons in `Animation State` control and buttons in `Animation Pose` control such as `new Pose`, `Open Pose` will be enabled after an animation state has been created. Those buttons are helpful when users frequently save their art work in case of loss of sprite data.

Create an Animation Pose inside an Animation State

Inside a created **Animation State**, there may be many **Animation State Pose** share an identical pose with same pose property like `image`, `duration`. Therefore, a small data set of a folder called **poses** is used for store identical poses together for each sprite. Instead of creating **Animation State Pose** for every single pose, users could only add poses from data set if existed. Users only create new pose and store it into the data folder **poses**, and this new created can be used and shared by many poses in **Animation State Pose** List.

1. Click the `New Animation State Pose` button in **Animation State Pose** Control.

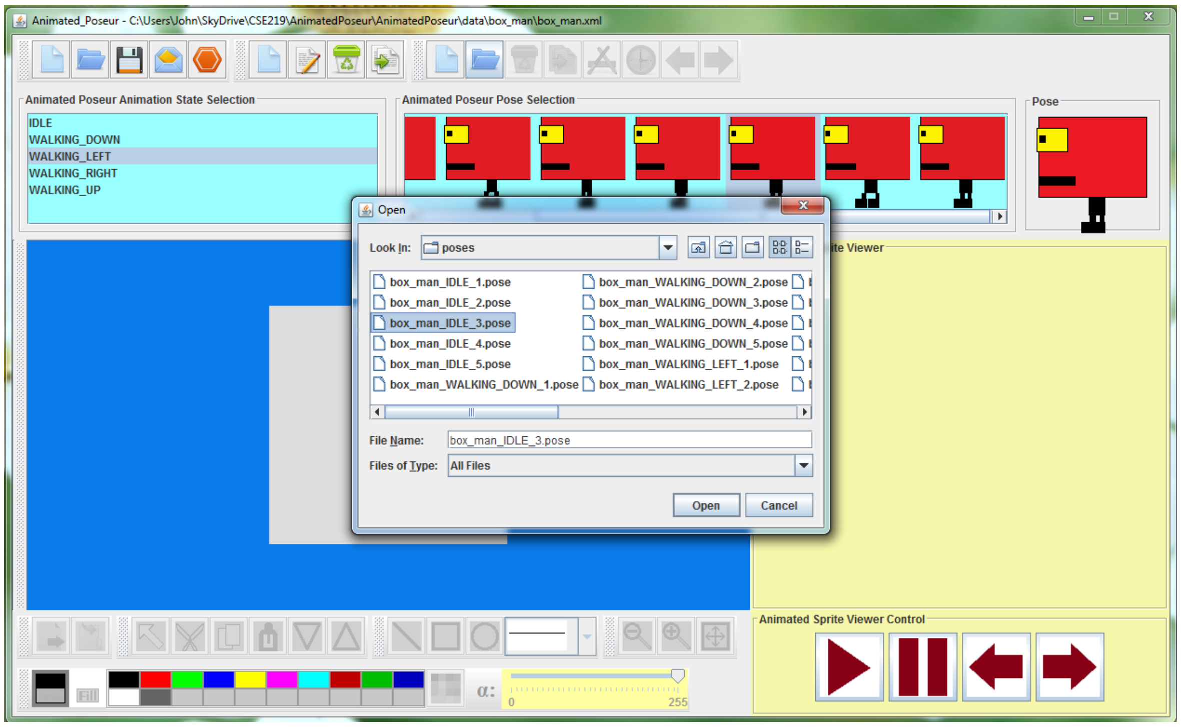
2. Draw the poses on painting canvas.

   Important Note – Users are *not* able to edit **Animated Sprite**, **Animated State** and **Animation State Pose** during the middle of painting. All works *must* be saved or discarded then actions can be continued.

3. Click `Save and Export to Image` button to save user's painting after finished or `Exit Pose Editor` if users only want to quit without saving their painting.

4. Click the `Add Animation State Pose` button in **Animation State Pose** Control. Select the file that being inserted. Then this file will be inserted to the tail of current **Animated Sprite Pose** list. Insertion of Animation State Pose is described as in Figure 7.1.

   Important Note – Poses are dynamic chained and they can be modified by order and displaying time. Other functions are available in this program such as *pose deletion*, *pose duplication*, *displaying time set* and *changing order of poses* as described in Figure 7.2.


Animated_Poseur - C:\Users\John\SkyDrive\CSE219\AnimatedPoseur\AnimatedPoseur\data\box_man\box_man.xml
Animated Poseur Animation State Selection
IDLE
WALKING_DOWN
WALKING_LEFT
WALKING_RIGHT
WALKING_UP
Animated Poseur Pose Selection
Pose
Open
Look In:
poses
Open
Cancel
Animated Sprite Viewer Control
Fill
α:
0
255

box_man_IDLE_1.pose
box_man_IDLE_2.pose
box_man_IDLE_3.pose
box_man_IDLE_4.pose
box_man_IDLE_5.pose
box_man_WALKING_DOWN_1.pose
box_man_WALKING_DOWN_2.pose
box_man_WALKING_DOWN_3.pose
box_man_WALKING_DOWN_4.pose
box_man_WALKING_DOWN_5.pose
box_man_WALKING_LEFT_1.pose
box_man_WALKING_LEFT_2.pose

File Name: box_man_IDLE_3.pose
Files of Type: All Files

Figure 7.1, Open an Animation State Pose from folder

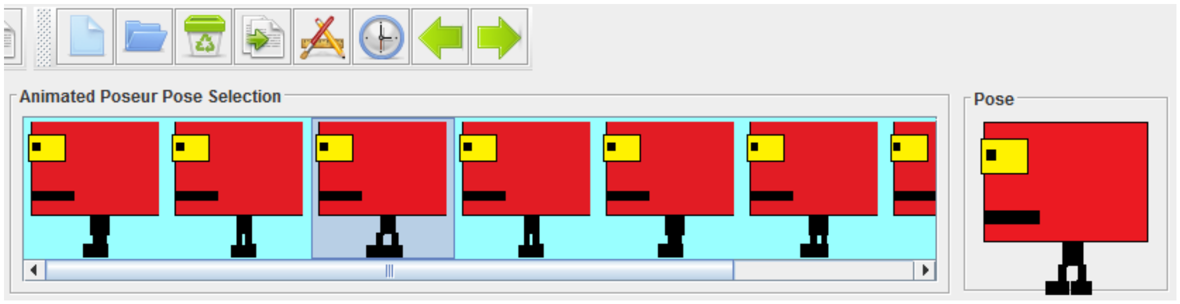


Figure 7.2 Pose selection and functions related to pose control

Edit an Animation State Pose

1. Click either `new animation state pose` button or edit `animation state pose` button to enter the mode of pose editing. The main purpose is for modification of the current pose.

2. Draw some shapes which are composed of your sprite. In **Animated Sprite Editor**, only *line*, *ellipse* and *rectangle* can be drawn on canvas.

3. Modify your shape by selecting the shape, moving around to an appropriate position and filling up the shape with any color. Note there are several additional features include *cut/copy/paste, changing order of top shape, changing the outline color and thickness, increasing/decreasing the canvas size* and *changing the transparency of a shape*.

4. Click `save and export to image` button to save your work. Otherwise, click `exit` button to quit.

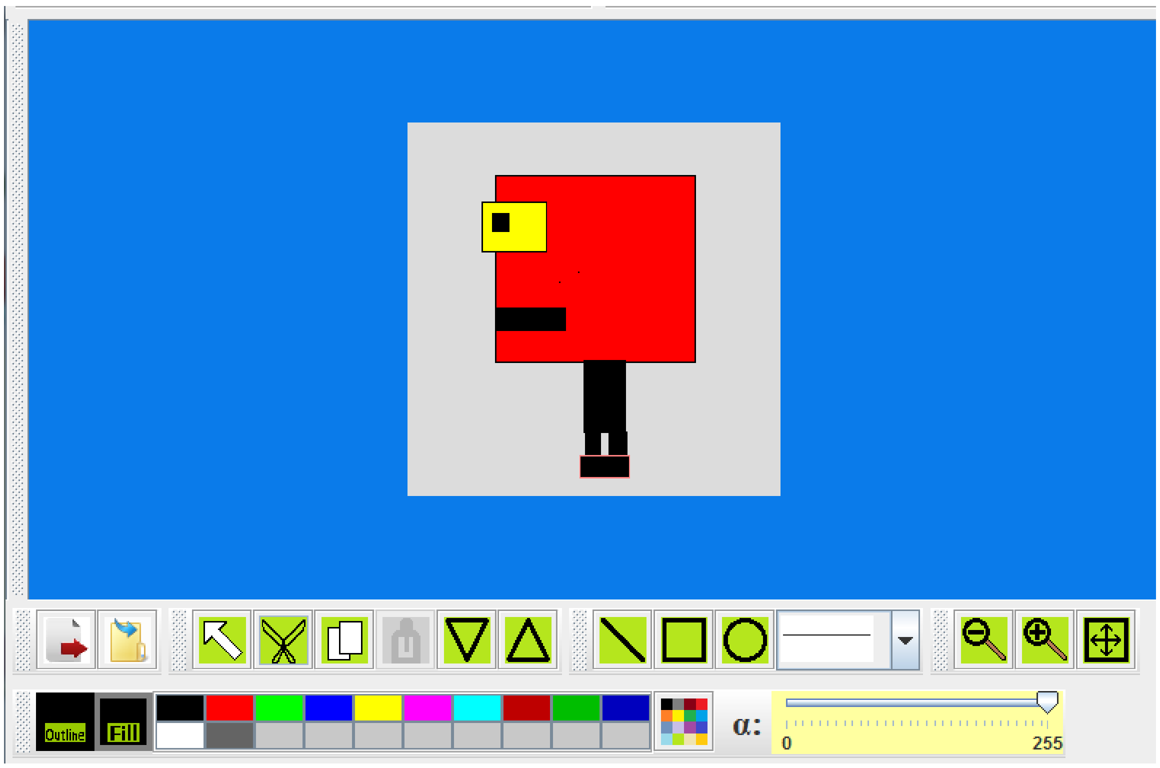


Figure 8, Draw a sprite on canvas

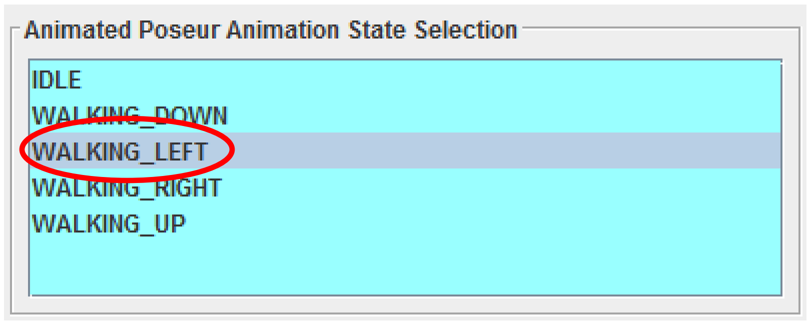
How to View Animation of Sprite

1. Click animation you want to render **Animation State** list. This list is a JList in Java as described in Figure 8.2

2. Click `Start Animation` button on yellow *animation sprite viewer*. You can `Pause Animation`, `Speed-up Animation` or `Slow-down Animation` by clicking button on right hand side of `Start Animation` button described in Figure 8.2.

3. Start over at step 1 if you want to take a look at other animation state.

Important Note – Do *not* try to close animation viewer because there is no option for leaving viewer without quitting the program. You might lose all your data if you accidently quit of program by clicking quit button at upper right corner without saving data.



Animated Poseur Animation State Selection

IDLE
WALKING_DOWN
WALKING_LEFT
WALKING_RIGHT
WALKING_UP

Figure 8.1 Animation state selections

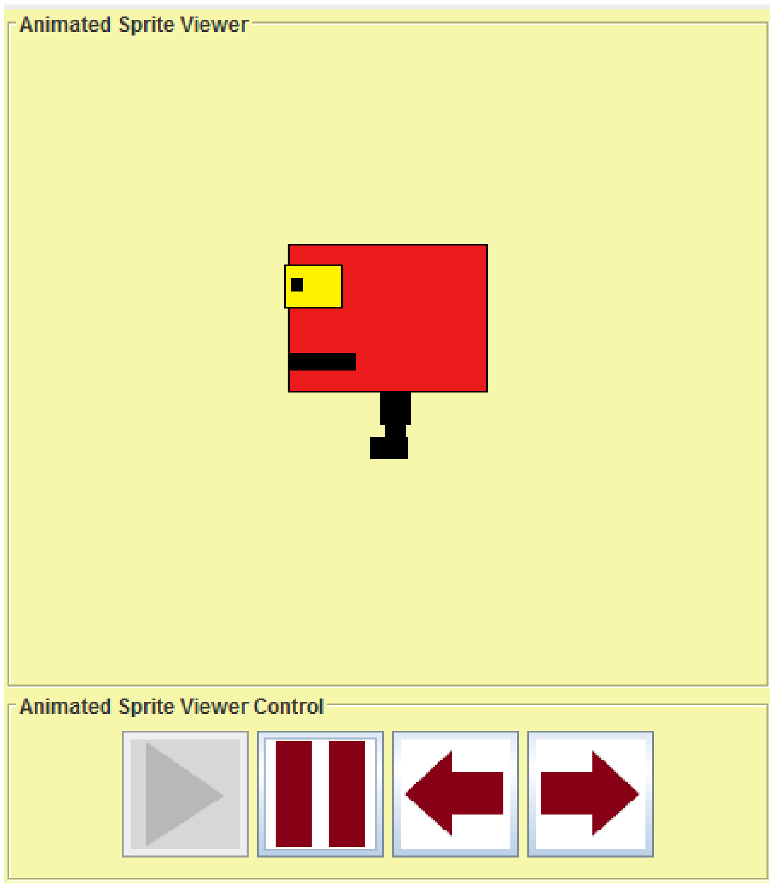


Figure 8.2, Animation viewer

For further information, contact the developer, Yunlong Zhang:
Yunlong.zhang@stonybrook.edu