Курсов Проект - Компютърна Графика

Графична симулация на стек и опашка чрез JOGL

+

Изготвил:
Марио Игнатов ф.н. 121209056 гр. 41
Христо Стайков ф.н. 121209100 гр. 41

Цел на проекта

Проектът е предназначен да представя визуално основните операции извършвани от 2-те най-използвани колекции в програмирането- Стек и Опашка. Главна задача е да се покаже силата на Java технологията, както и лекотата на използване на OpenGL с езика.

Техническо описание

- Чрез разработеното приложение, потребителят може да симулира добавянето и изваждането на елементи в стек или опрашка. В основния екран може да се види анимираното поведение на програмата, а в дясната част - нагледно какво се случва в колекцията. За тази цел има поле за въвеждане на стойност на всеки един елемент в колекцията.
- Разработката е осъществена чрез **Java OpenGL** технологията. Като работна среда са използвани **NetBeans** и **Eclipse**. Нужните библиотеки са:

jogl.jar - свързва Java с OpenGL API-то. Предоставя хардуарна поддръжа на 3D приложения, писани на Java.

gluegen-rt.jar - библиотека, която генерира Java код, който е нужен за достъп до някои библиотеки в OpenGL, които са писани на C.

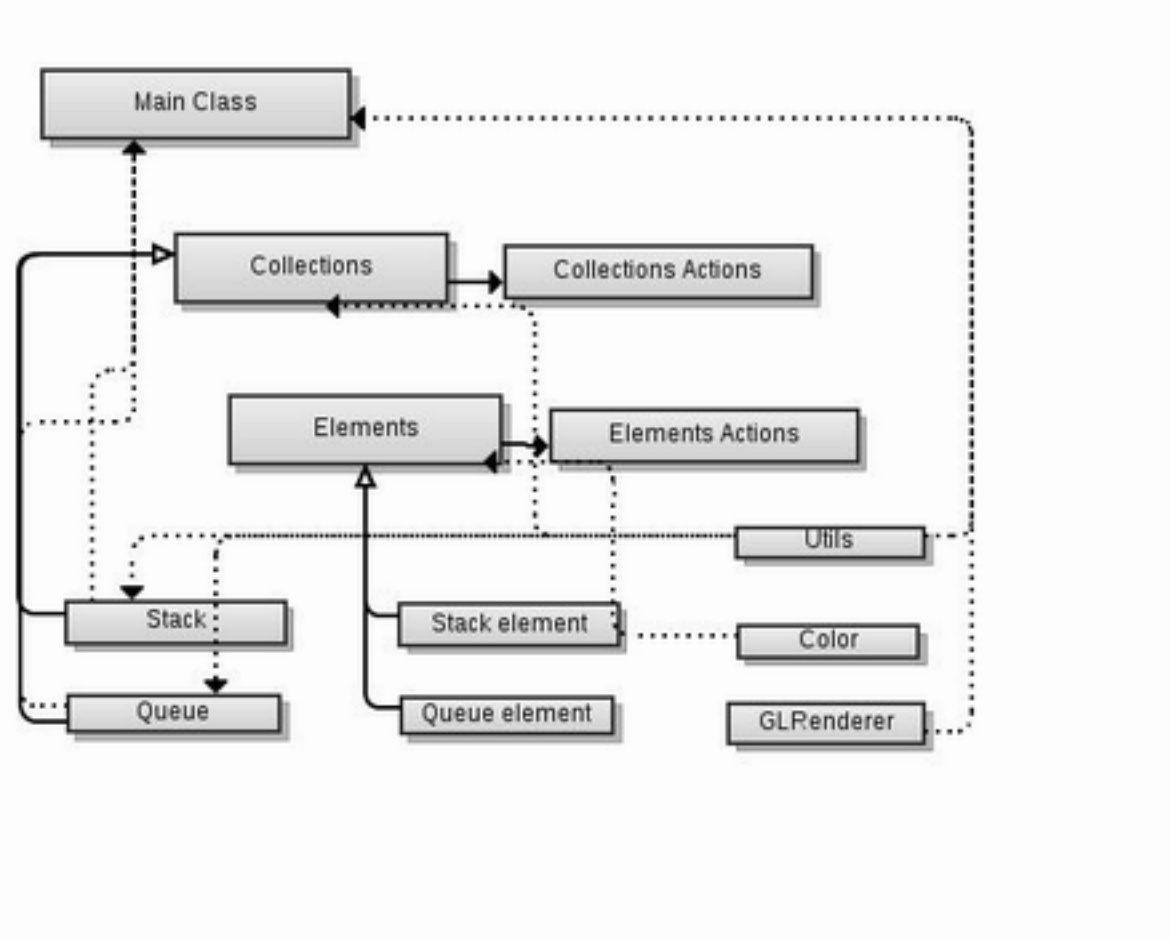
- Текущото състояние на приложението е реализирано на база на други 3rd party по-малки проекти. **Основната версия беше създадена като курсов проект във 2-ри курс по Java. Поради тази причина, част от функционалността е скрита в текущата версия но фигурира в кода.**
- Основните използвани методи и алгоритми са:
  -Транслация на обект - **glTranslatef**,
  -Вход и изход на матрицата от стека - **glPushMatrix** и **glPopMatrix**
  -Създаване на сфера чрез glut - **glutWireSphere** / **glutSolidSphere**
  -Оцветяване - **glColor3f**
  -Добавяена на “ефекти” към сцената - **glEnable**
  -Очертаване на върхове на фигура (куб в случея) - **glVertex3f**
  -Както и други....

Описание на програмната реализация

Графичният интерфейс на приложението е създаден чрез **SWING** през **NetBeans**. Основната роля играе **Canvas** обекта, в който се случва цялата

анимация. Останалите бутони и форми са стандартни компоненти. Основната логига на графичние интерфейс и компонентие е в **MainSimulationWindow.java**.

Backend реализацията се състои от няколко класа чрез следната йерархия:

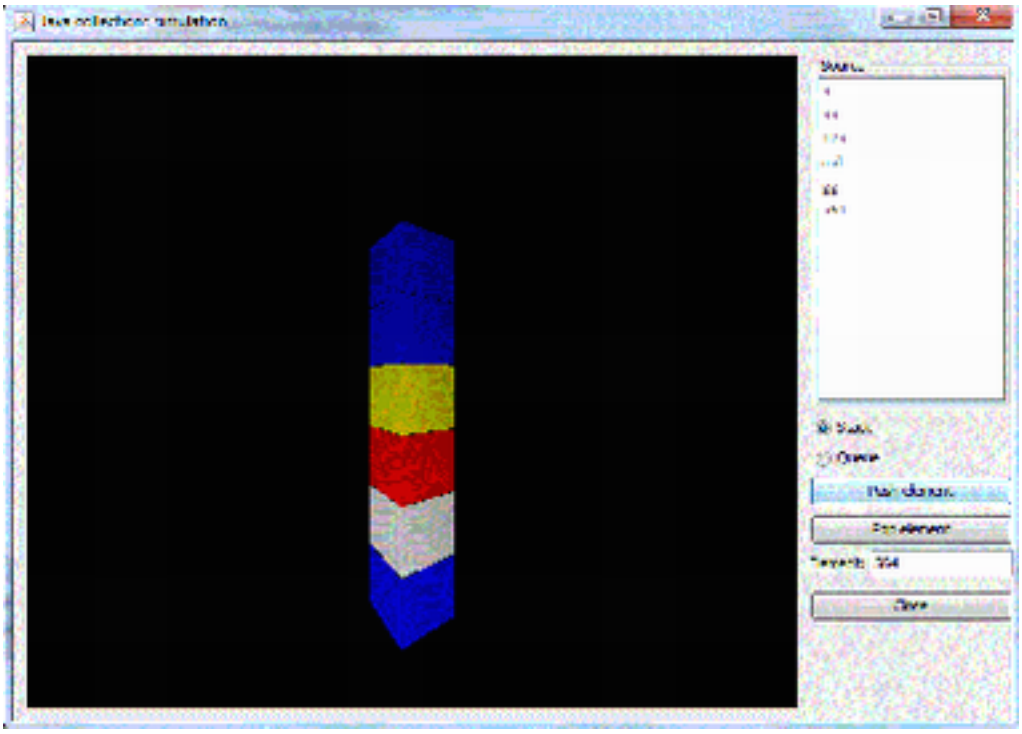


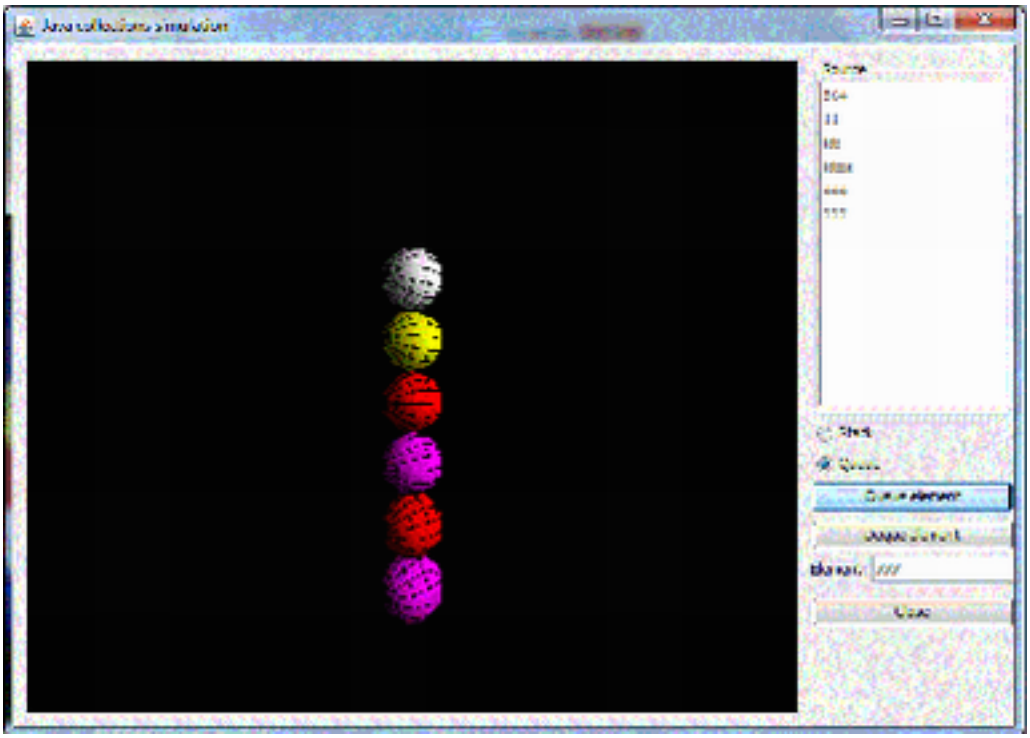
Съществуват два интерфейса - **Collection Actions** - предоставящ фукнционалност на колекцията. В него са дефинирани методите за изрисуване на обект - **draw,** добавяне и изваждане на елемент от колецията - **addElement**, **removeElement**.
Интерфейсът **ElementsActions** съдържа единствен метод **draw.** Тези интерфейси се имплементират от **двата базови класа Collections и Elements.** В тях са дефинирани някои основни променливи/константи като: координати, цветове, статус на текущия елемент, стойности за скорост, ъгъл на въртене, ускорение, изместване, размери и др. . Съществувавт и два помощни класа - **Color** и **Utils**, в които, съответно, дефинират цвят и някои помощни изброени типове.
От гледна точка на компютърната графика важна роля играя класът **GLRenderer.** Той върши основната работи при показването на анимацията на екрана. Важно методи там са:

Init() - Създаването на редата. Настройка на свойствата и поведението на анимацията. Дефинират се пропъртита като: източник на светлина, сенки, гладкост на повърхността и други.
reshape() - Създаване на обекти. Извикване на нужните матрици, добавяне на перспектива, региона на показване от гледна точка на компюътра.
display() - Изображение на обкетите на екрана на потребителя. Настройка на потребителския изглед и изчистване на ненужните матрици.

При **StackElement** има метод **drawCube(),** който създава обект куб по зададени върхове. Метод **draw(),** който спрямо статута на текущия обект изрисува анимацията за добавяне или за премахване на куб от стека. **drawDestruction(), -** анимацията за премахване на обект. Задават се параметри за въртене, вика се метод от GL библиотеката, която завърта обекта, изрисува се на ново с новата настройка на gl-а и при определено условие се маркика като унищожен. Аналогично е и при **QueueElement.**
Когато един е в статус **DESTROYED** в базовия клас Collection се прави проверка за него, и ако е така елемента се премахва от колекцията. Когато даден елемент бива отбелязан като унищожен в класът на съответна колекция в елемента от този тип се прави проверка при изрисуването (**draw()**) за това дали трябва да се конфигурира **gl** обекта за добавяна или премахване на елемент - **MARKED_FOR_DESTRUCTION**.
Обектите се заключват по време на анимацията, за да не може да се извиква многократно някоя операция и да се препълни стека с матрици за визуализиране. Всичко това се извършва в една нова нишка при **MainSimulationWindow.**

Примерни екрани





За съжаление, анимацията при премахване на обект не може да бъде показана.

Финализация

Като обобщение, JOGL е сравнително мощна технология. Не е много разпространена защото има много по-добри варианти за 3D компютърна графика, като стандартния OpenGL чрез C. Въпреки това предоставя на разработчиците с въображение еднин избор на работа. Първоначалната идея на този проект беше като обучителна програма за представяне на работата на двете колекции. Основните точки са покрити, но за напред може да се добави нова функционалност като хеш таблици, речници и тн.. Другия идея е, да се разработи версия като за OpenSource продукт, който да служи не само за обучение на хора запознаващи се с основите на програмирането, а и на такива, които да продължат работа по него за развитие на личните качества като програмисти на JOGL.