**Сопроводительный лист на издание в авторской редакции**

Название работы Интерактивная анимация нескольких объектов Методические указания к выполнению лабораторной работы № 23 для студентов очной формы обучения по направлениям подготовки 09.03.04 – «Программная инженерия», 09.03.01 «Информатика и вычислительная техника», 02.03.03 «Математическое обеспечение и администрирование информационных систем»

Актуальность и соответствующий   
научно-методический уровень подтверждаю \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись научного редактора)

Рукопись сверена и проверена автором \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись автора)

Рекомендуется к изданию \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись заведующего кафедрой)



**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

Брянский государственный технический университет

Утверждаю

Ректор университета

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.Н. Федонин

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017

ПРОГРАММИРОВАНИЕ

ИНТЕРАКТИВНАЯ АНИМАЦИЯ НЕСКОЛЬКИХ ОБЪЕКТОВ

**Методические указания**

**к выполнению лабораторной работы № 23**

**для студентов очной формы обучения**

**по направлениям подготовки**

**09.03.04 – «Программная инженерия»,**

**09.03.01 «Информатика и вычислительная техника»,**

**02.03.03 «Математическое обеспечение и администрирование информационных систем»**

**Брянск 2017**

**УДК 004.**

Интерактивная анимация нескольких объектов [Текст] + [Электронный ресурс]: методические указания к выполнению лабораторной работы № 23 для студентов очной формы обучения по направлениям подготовки 09.03.04 – «Программная инженерия», 09.03.01 «Информатика и вычислительная техника», 02.03.03 «Математическое обеспечение и администрирование информационных систем». – Брянск, 2017. – 7 с.

Разработали:

Д. И. Булатицкий

канд. техн. наук, доц.,

Е..В. Коптенок

ассистент

Рекомендовано кафедрой «Информатика и программное обеспечение» БГТУ

**Методические издания публикуются в авторской редакции**

# Цель работы

Цель работы – научиться создавать интерактивную анимацию нескольких объектов при помощи библиотеки SDL.

Продолжительность работы – 2 часа.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Для сохранения каждого полученного сообщения в общем виде и последующего анализа используется объединение типа **SDL\_Event.**

Функция **SDL\_PollEvent** вызывает функцию принудительного обновления очереди событий (однократный опрос возможных источников событий – функция **PumpEvents,** не имеющая аргументов и не возвращающая никакого значения), затем проверяет, имеется ли в очереди хотя бы одно событие любого типа, ожидающее обработки. Если очередь пуста, функция **SDL\_PollEvent** возвращает 0. Если очередь не пуста и параметр event не равен NULL, то очередное (первое) событие извлекается из очереди и сохраняется в объединении **SDL\_Event**.

Для символического представления констант, описывающих разные типы событий, в SDL используется перечисление SDL\_EventType. Рассмотрим основные типы событий, обозначая их здесь и далее соответствующими константами из данного перечисления (то есть, «событие SDL\_KEYDOWN» – событие, при котором поле type объединения SDL\_Event имеет значение, равное значению константы SDL\_KEYDOWN):

SDL\_ACTIVEEVENT – приложение стало активным (с которым работает пользователь) или перестало быть активным;

SDL\_KEYDOWN – нажата клавиша на клавиатуре;

SDL\_KEYUP – отпущена клавиша на клавиатуре;

SDL\_MOUSEMOTION – перемещена мышь;

SDL\_MOUSEBUTTONDOWN – нажата клавиша мыши;

SDL\_MOUSEBUTTONUP – отпущена клавиша мыши;

SDL\_QUIT< – запрос выхода из программы по действию пользователя (например, по нажатию мышью системной кнопки закрытия окна);

SDL\_VIDEORESIZE – пользователь изменил размер окна и требуется изменение видеорежима.

Полный список событий представлен на официальном сайте: https://wiki.libsdl.org/SDL\_EventType.

Коды клавиш клавиатуры представлены по ссылке на официальный сайт SDL: https://wiki.libsdl.org/SDL\_Scancode

# ХОД РАБОТЫ

**Задание1:** пусть в окне расположены несколько шаров. При нажатии по шару мышью, он исчезает. В правом верхнем углу отображается количество удаленных шаров.

Работа выполнена в Visual Studio Community 2015.

Создадим пустой проект и добавим в него файл Source.cpp.

Подключим библиотеку SDL. Произведем щелчок правой кнопкой мыши по проекту в обозревателе решений и выберем «Управление проектами NuGet» (Рис.3.1.):

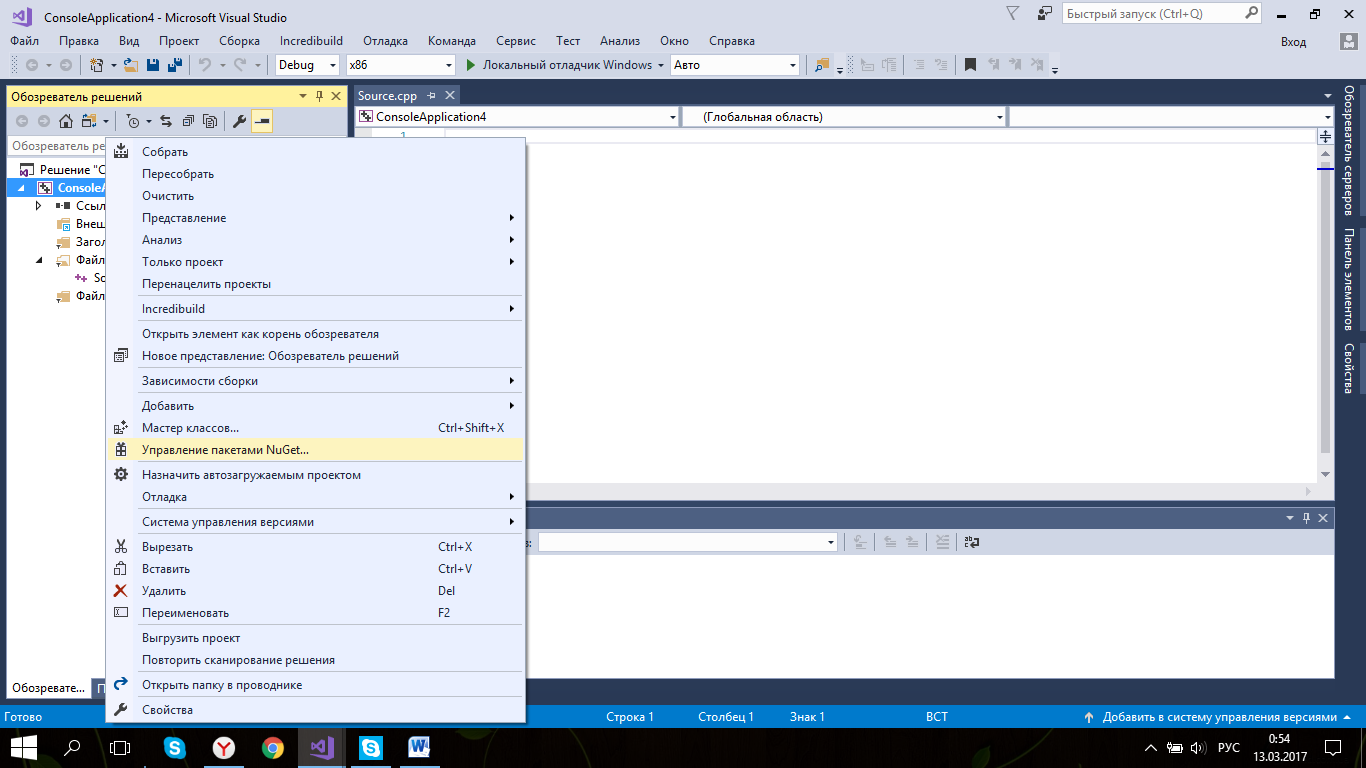


Рис.3.1. Пункт меню «Управление проектами NuGet»

Осуществим поиск, как показано на Рис.3.2.

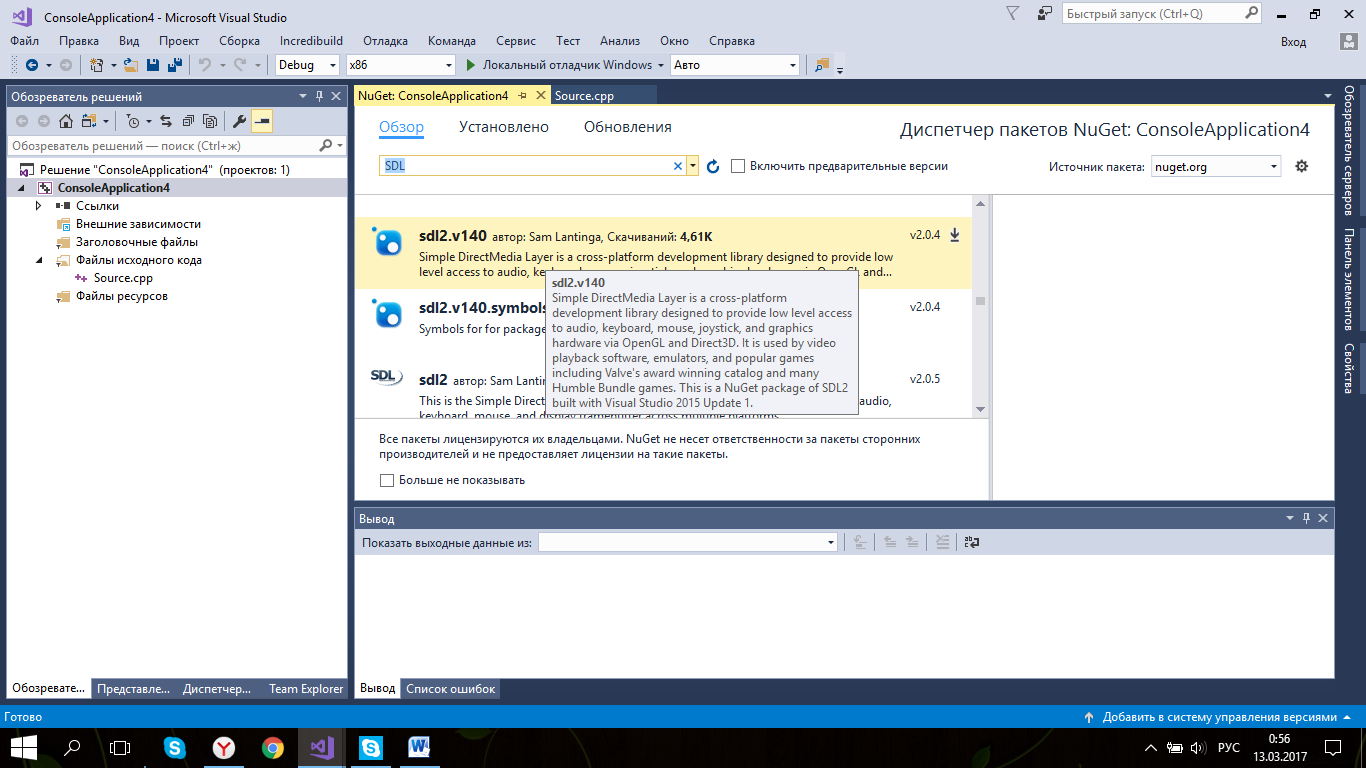


Рис.3.2. Поиск библиотеки

Осуществим установку, как это показано на Рис.3.3.

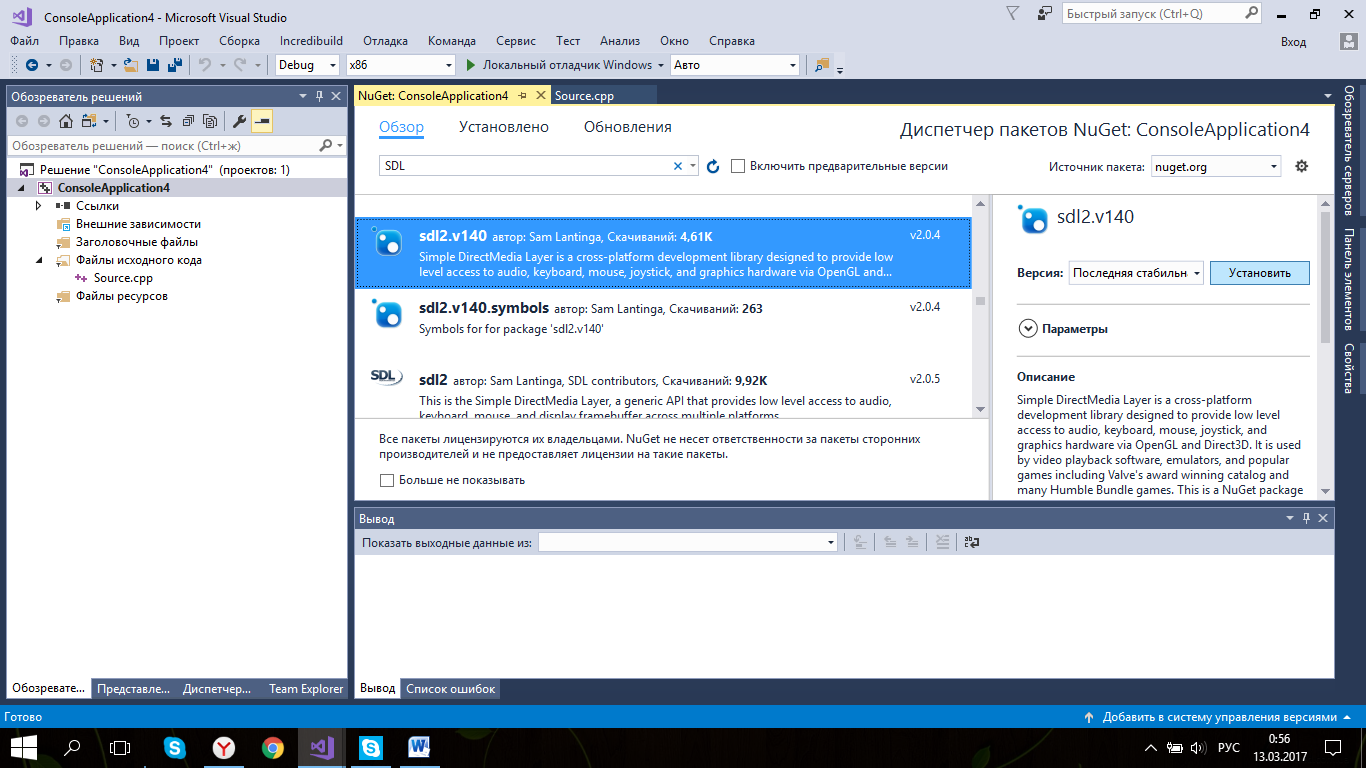


Рис.3.3. Установка библиотеки

Перейдем к файлу с исходным кодом. Инициализируем графический режим, вызовем отображение окна. Для отображения русскоязычного заголовка перед текстом укажем «u8». Добавим обработчик события закрытия окна внутри игрового цикла и освобождение памяти.

Код программы приведен на Рис.3.4.

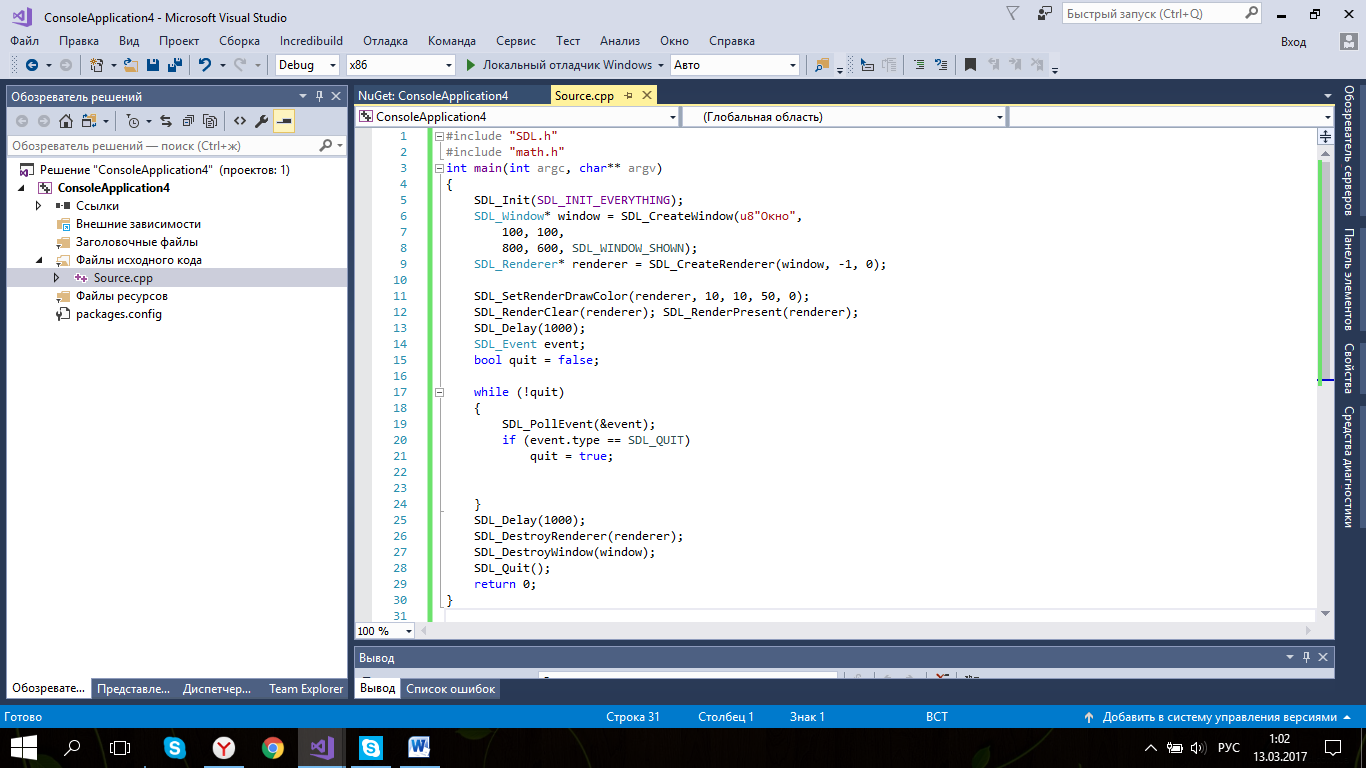


Рис.3.4. Код программы

Напишем функцию отображения текста.

Пропишем подключение заголовочного файла: #include <SDL\_ttf.h>.

Подключим библиотеку, как это показано на Рис.3.5.

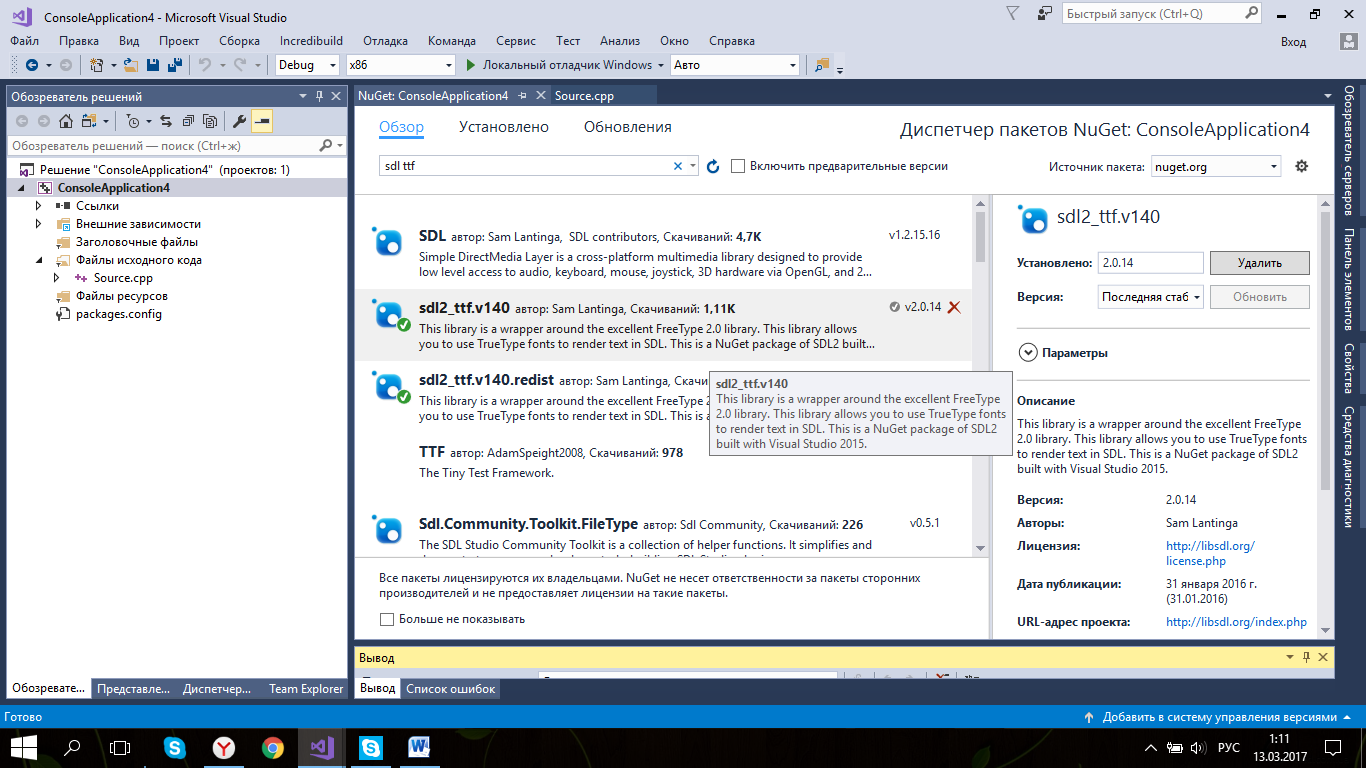


Рис.3.5. Подключение библиотеки для работы со шрифтами

В случае возникновения ошибки добавим в папку Debug проекта библиотеку freetype262d.dll.

Создадим функцию, которой будем передавать окно, адрес рендера и текст:

void draw\_text(SDL\_Window\* window, SDL\_Renderer\* &renderer, char\*text) {}

В ней разместим следующий код:

TTF\_Init();

TTF\_Font\* my\_font =TTF\_OpenFont("Text.ttf", 100);

SDL\_Color fore\_color = { 130,140,50 };

SDL\_Color back\_color = { 188,155,166 };

SDL\_Surface\* textSurface = NULL;

textSurface = TTF\_RenderText\_Shaded(my\_font, text, fore\_color, back\_color);

Первая строка инициализирует режим работы с текстом. Далее загружается шрифт (файл должен быть добавлен в проект), задаются цвет шрифта и фона, создается рабочая поверхность, в которую заносится текст.

Далее задаются координаты текста, создается текстура, в которую преобразуется поверхность. Данная текстура передается в рендер для отрисовки. В заключении освобождается память.

SDL\_Rect rect = { 0,0, 30, 30 };

SDL\_Texture \*texture = SDL\_CreateTextureFromSurface(renderer, textSurface);

SDL\_RenderCopy(renderer, texture, NULL, &rect);

TTF\_Quit();

Опишем функцию рисования шара:

void draw\_ball(SDL\_Renderer\* &renderer, SDL\_Rect rect)

{

SDL\_Surface \*myImage = SDL\_LoadBMP("1.bmp");

SDL\_SetColorKey(myImage, SDL\_TRUE, SDL\_MapRGB(myImage->format, 255, 255, 255));

SDL\_Texture \*texture = SDL\_CreateTextureFromSurface(renderer, myImage);

SDL\_RenderCopy(renderer, texture, NULL, &rect);

}

На вход будут передаваться координаты и размер прямоугольной области, в которой будет производиться рисование и рендер.

Внутри функции производится загрузка изображения, установка прозрачности, преобразование поверхности с изображением в текстуру и наложение текстуры на рендер.

Внутри главной функции программы создадим массив прямоугольников, задающих положение шаров, и циклически выведем изображения на экран. Цикл должен располагаться до игрового цикла:

SDL\_Rect \*ball = new SDL\_Rect[5];

for (int i = 0; i < 5; i++)

{

ball[i] = { i\*50 + 110,i\*50 +10 ,50+10\*i ,50 +10\*i};

draw\_ball(renderer, ball[i]);

}

SDL\_RenderPresent(renderer);

Теперь нам необходимо написать обработчик щелчка левой кнопкой мыши. Необходимо проверить попадание координат мыши в прямоугольник с шаром. В случае выполнения условия размеры прямоугольника уменьшатся до нулей, экран очистится и прямоугольники перерисуются снова. Данный код должен содержаться в игровом цикле:

if (event.button.button==SDL\_BUTTON\_LEFT)

for (int i = 0; i < 5; i++)

{

if (event.button.x >= ball[i].x && event.button.x<=ball[i].x + ball[i].w && event.button.y>=ball[i].y && event.button.y <= ball[i].y + ball[i].h)

{

ball[i].w = 0;

ball[i].h = 0;

SDL\_RenderClear(renderer);

for (int j = 0; j < 5; j++)

{

draw\_ball(renderer, ball[j]);

}

SDL\_RenderPresent(renderer);

}

Осталось осуществить подсчет удаленных шаров. Внутри main() заведем целочисленную переменную int k=0; которую будем увеличивать каждый раз, когда будет выполняться условие нажатия на шарик. Также заведем строковую переменную char \*text=new char[10];

До начала игрового цикла выведем количество очков, равное нулю. Преобразуем целочисленное значение переменной k в строку (потребуется #include"stdlib.h"):

itoa(k, text, 10);

Вызовем функцию отрисовки текста для полученного значения:

draw\_text( window, renderer,text);

SDL\_RenderPresent(renderer);

В случае выполнения условия после увеличения счетчика аналогично выведем счет:

itoa(k, text, 10);

draw\_text(window, renderer, text);

SDL\_RenderPresent(renderer);

Результат запуска программы представлен на Рис.3.6.:

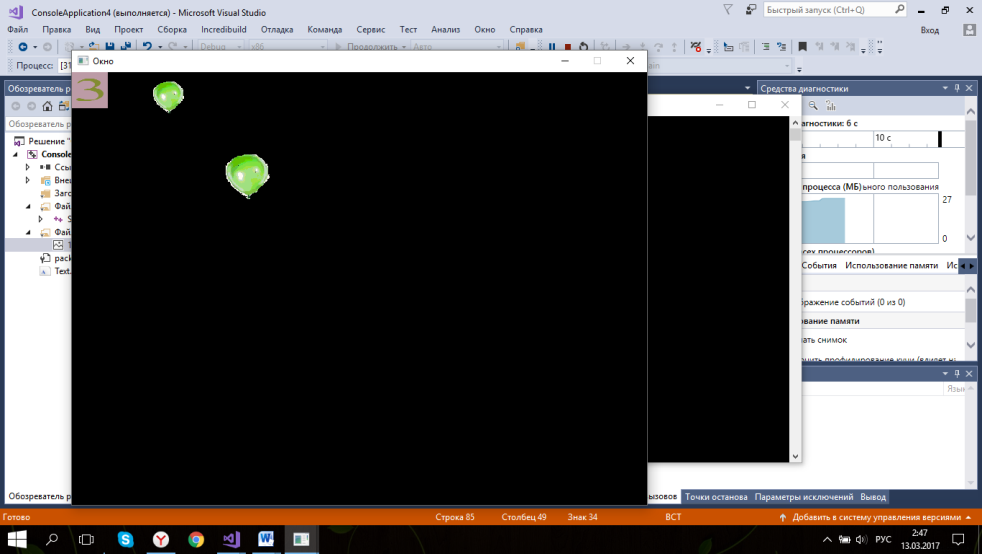


Рис.3.6. Результат запуска программы

# ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

1. Модифицировать программу из Задания 1 таким образом, чтобы шары отображались в случайной позиции, а за их удаление назначалось различное количество очков. Предусмотреть вывод стоимости удаления на шаре, использовать собственные текстуры.
2. По вариантам (первая часть задания обязательна, вторая – дополнительная; должен быть реализован счетчик очков):
3. После удаления появляется новый шар другого размера. Предусмотреть невозможность наложения шаров друг на друга
4. Шары не статичны, появляются по очереди и падают сверху вниз. Предусмотреть возможности ловить их полосой, без возможности поднимать ее с нижнего края экрана (как в арканоиде, но шары не отскакивают)
5. Шары статичны, но есть объект, передвигаемый клавиатурой по окну и удаляющий шар при столкновении. Предусмотреть возможность появления нового шара с другими характеристиками и внешним видом, но недоступного для удаления.
6. Шары статичны, но щелчком мыши выбирается шар, который получает способность передвигаться при нажатии стрелок на клавиатуре; его цвет меняется; другие шары при столкновении с ним удаляются. Другие шары подвижны; столкновения с другими шарами и границей обрабатываются.
7. Шары постоянно меняют свое местоположение. Время смены местоположения шаров различно.
8. Шары движутся по круговой орбите с различным радиусом. Размер шаров постоянно меняется.

# КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего предназначена библиотека SDL?
2. Из каких этапов состоит создание простейшего приложения с использованием графики?
3. Из каких этапов состоит отображение простейшей анимации средствами SDL?
4. В чем заключается принцип разработки интерактивной анимации средствами SDL?

# Список рекомендуемой литературы

## Основная литература

1. Павловская Т.А. C#. Программирование на языке высокого уровня:учеб. Для вузов.-м. [и др.]:Питер,2014.-432
2. Давыдова Н.А.,боровская Е.В. Программирование: учеб. Пособие. М.:Бином. Лаб. Знаний,2011.-238 с.
3. Булатицкий Д.И. Объектно-ориентированное программирование: учеб. пособие.- Брянск: изд-во БГТУ,2008.-120 с.

## Дополнительная литература

1. Тюльпинова Н.В. Алгоритмизация и программирование: учеб. Пособие/Брян. Гос. Техн. Ун-т.- Брянск:изд-во БГТУ,2012.-199 с.-[+ электронная копия].
2. Иванова Г.С. Программирование: основы алгоритмизации и процедурное программирование, объектно ориентированное программирование: учеб. Для вузов.-2-е изд., стер.-м.:Кнорус, 2014.-425 с.

Интерактивная анимация нескольких объектов [Текст] + [Электронный ресурс]: методические указания к выполнению лабораторной работы № 23 для студентов очной формы обучения по направлениям подготовки 09.03.04 – «Программная инженерия», 09.03.01 «Информатика и вычислительная техника», 02.03.03 «Математическое обеспечение и администрирование информационных систем». – Брянск, 2017. – 7 с.

БУЛАТИЦКИЙ ДМИТРИЙ ИВАНОВИЧ

КОПТЕНОК ЕЛИЗАВЕТА ВИКТОРОВНА

Научный редактор В.Я. Израилев

Компьютерный набор Е.В. Коптенок

Иллюстрации Е.В. Коптенок

Подписано в печать 20.02.2017. Формат 60х84 1/15 Бумага офсетная. Офсетная печать. Усл.печ.л. \_\_\_ Уч.-изд.л. \_\_\_ Тираж 1 экз. Бесплатно

Брянский государственный технический университет

Кафедра «Информатика и программное обеспечение», тел. 56-09-84

241035, Брянск, бульвар 50 лет Октября, 7 БГТУ, тел. 58-82-49