Курс «Основы программирования»

Власенко Олег Федосович
SimbirSoft

Сквозной проект

Создание (простой) игры

СП1. Выбор варианта игры. Отрисовка всех объектов.

СП2. Управление героем.

СП3. Делаем персонажей много

СП4. Сохранение/загрузка состояния.

СП5. Красота внутри и снаружи. Модули. Рекурсия

СП6. Финализация

Что такое «Сквозной проект» и зачем он вам?

Чтобы стать ИТ специалистом, необходимо не только изучить основы основ, но и получить опыт работы на достаточно больших проектах. Поэтому в рамках курса «Основ программирования» кроме простых лабораторных работ (и усложненных задач к этим лабораторным работам) есть еще Сквозной проект, в котором в течение семестра каждый студент получит опыт длительной разработки достаточно объемного проекта.

Сквозной проект выполняется СТРОГО ПО ЭТАПАМ.
Он может быть сделан быстро досрочно – и он будет зачтен, в случае если авторство будет бесспорно.
Но в случае сильного запаздывания по этапам этот проект (даже сделанный! И даже при бесспорном авторстве!!!!!) не будет засчитан!

В рамках Сквозного проекта каждый студент будет делать свой собственный (уникальный!!!) вариант **ИГРЫ**.

Какой вариант игры уже сделан для демонстрации?

Название игры: «Жизнь ёжика»

Объекты (Персонажи и предметы): Ёжик, грибы (его еда), яблоки (его лекарства), лиса (его враг), нож (его оружие)

Сценарий основной: Ёжик должен собрать все яблоки на этом уровне – тогда он переходит на следующий. Если он прошел все уровни – он победил.

Сценарии другие: Есть лиса - враг ёжика. Если он встретился с ней больше раз, чем следовало – игра проиграна. После встреч с лисой ёжик может подлечиться, если будет есть яблоки. Если ёжик обзаведется ножом, то при встрече с лисой не поздоровиться уже лисе.

Управление: Ёжик передвигается при нажатии клавиш – влево/вправо, вверх/вниз. Лиса перемещается автоматически – в сторону к ежику или случайно.

Какой вариант игры должен выбрать и реализовать студент?

Вам предстоит в самом начале выбрать идею для своей **игры**.

Эта идея должна быть
А) отличной от идей всех ваших одногруппников
Б) Всё, что будет сделано в игре, вы либо уже знаете как сделать на момент начала работы над игрой, либо вы изучите все необходимое в рамках курса «Основы программирования»
В) Всю игру вы должны сделать в достаточной степени самостоятельно. Допустимо консультироваться или заимствовать код, но

1) любая строчка кода вами понята и может быть легко (и быстро) объяснена
2) более 50% строк кода написано вами индивидуально без какой-либо помощи и заимствований.

Вы можете выбрать варианты из предоставленного списка игр. В этом случае гарантируется, что вы в состоянии будет реализовать эту игру в рамках этого семестра, и что все нужные для проекта знания включены в темы лекций и лаб работ.

Если вы придумываете свой собственный вариант, то вы обязаны его согласовать с вашим преподавателем по лаб работам. Если он его одобрит – лишь после этого вы можете приступать к реализации.

Какие требования к Сквозному проекту?

1) Проект выполняется на Си в среде Microsoft Visual Studio
2) Проект выполняется персонально. Кода, написанного лично, должно быть не менее 50%
3) Каждая строка позаимствованного кода (в том числе и сгенерированного VS, созданного в кооперации с консультантами, найденного в интернете и т.п.) должна быть понятна сдающему. (Понятность проверяется как опросом, так и заданием что-то в этом коде изменить)
4) Сквозной проект является ДОПОЛНИТЕЛЬНЫМ проектом. В случае его отсутствия у студента, студент имеет возможность получить положительную оценку за дисциплину.
5) Сквозной проект выполняется либо четко по этапам, либо с опережением этапов. Если студент отстает от графика и группы, то проект ему не засчитывается. Совсем не засчитывается ☹ (это делается как для того, чтобы студенты привыкали к четкому графику, так и для уменьшения возможностей списывания)

Варианты игры

Подход 1. Изучите реализованную игру «Жизнь ёжика». Подумайте, какие герои могут быть в вашей версии игры? Какой сценарий будет у вас? Например, можно сделать игру (по аналогии с «Жизнью ёжика») где «Девочка» собирает «Цветы», чтобы сплести «Венок». Объект «Крапива» представляет для нее определенную угрозу. А если она догонит «Бабочку», то получает бонус.
После того как придумаете игру, опишите все объекты и сценарии в виде текста. И согласуйте вариант вашей игры с преподавателем ведущим лаб работы.

Подход 2. Выберите готовый вариант из списка
(СПИСОК В РАЗРАБОТКЕ. В настоящий момент он недоступен для выбора вариантов!)

Сквозной проект – этап 1 (СП1)

Выбор варианта игры.
Отрисовка всех объектов.
(Статическая картинка).
Презентация итогов этапа 1 преподавателю.

Задачи ЭТАПА 1

Задача 1. Выбор темы игры
Задача 2. Утверждение темы игры
Задача 3. Рисуем все персонажи и предметы в виде картинки (в компьютере или на бумаге)
Задача 4. Создать приложение, содержащее все объекты из игры
Задача 5. Презентация преподавателю получившегося приложения на занятии

Задача 1. Выбор темы игры

Вам необходимо выбрать или придумать тему для игры и оформить ее в виде текста. В описание вашего варианта рекомендуется придерживаться следующей структуры.

Название игры: *«Жизнь ёжика»*

Объекты (Персонажи и предметы): *Ёжик, грибы (его еда), яблоки (его лекарства), лиса (его враг), нож (его оружие)*

Сценарий основной: *Ёжик должен собрать все яблоки на этом уровне – тогда он переходит на следующий. Если он прошел все уровни – он победил.*

Сценарии другие: *Есть лиса - враг ёжика. Если он встретился с ней больше раз, чем следовало – игра проиграна. После встреч с лисой ёжик может подлечиться, если будет есть яблоки. Если ёжик обзаведется ножом, то при встрече с лисой не поздоровиться уже лисе.*

Управление: *Ёжик передвигается при нажатии клавиш – влево/вправо, вверх/вниз. Лиса перемещается автоматически – в сторону к ежику или случайно.*

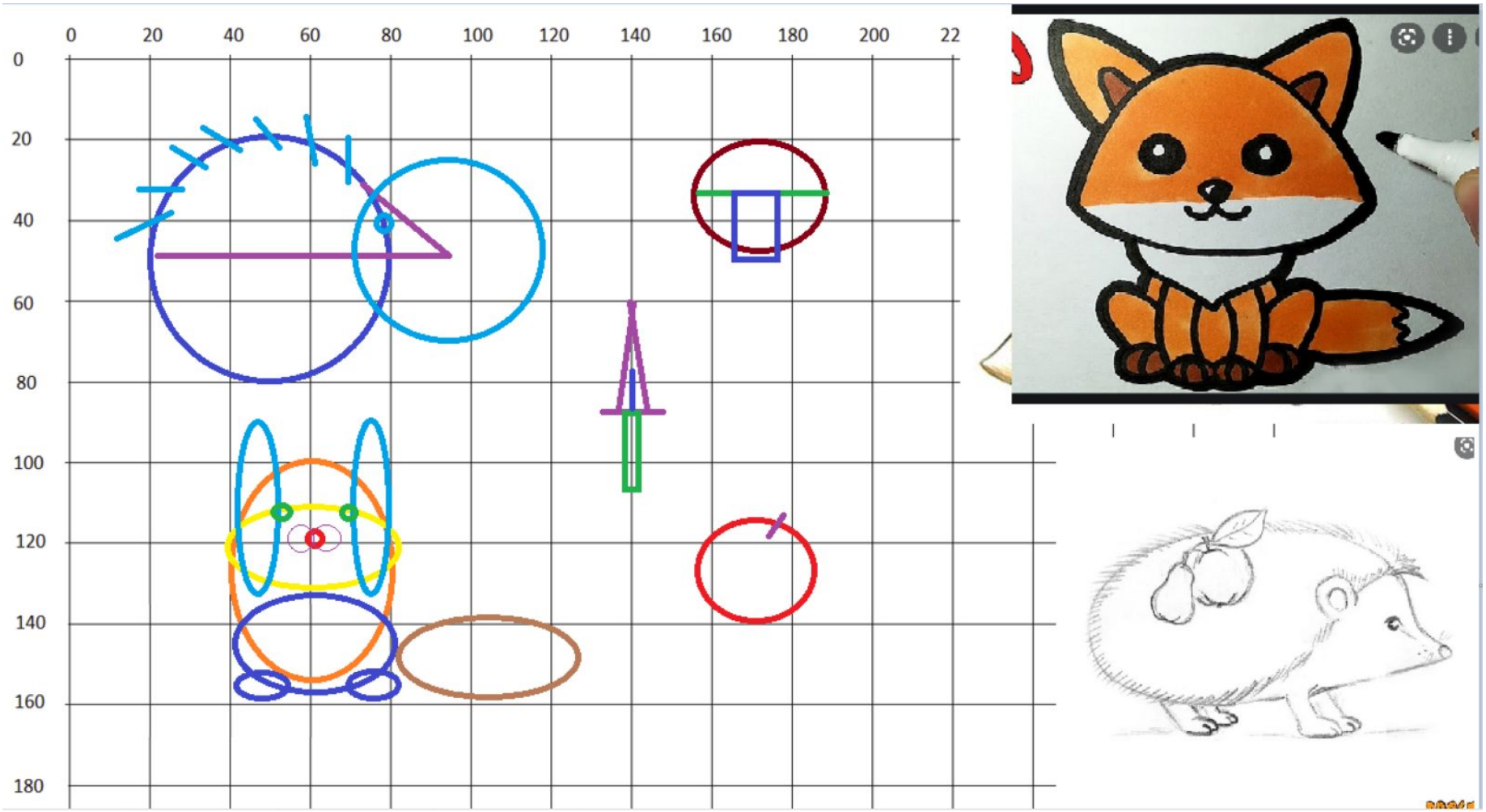
Задача 2. Утверждение темы игры

После выбора темы вам необходимо утвердить её у вашего преподавателя, ведущего лабораторные работы.

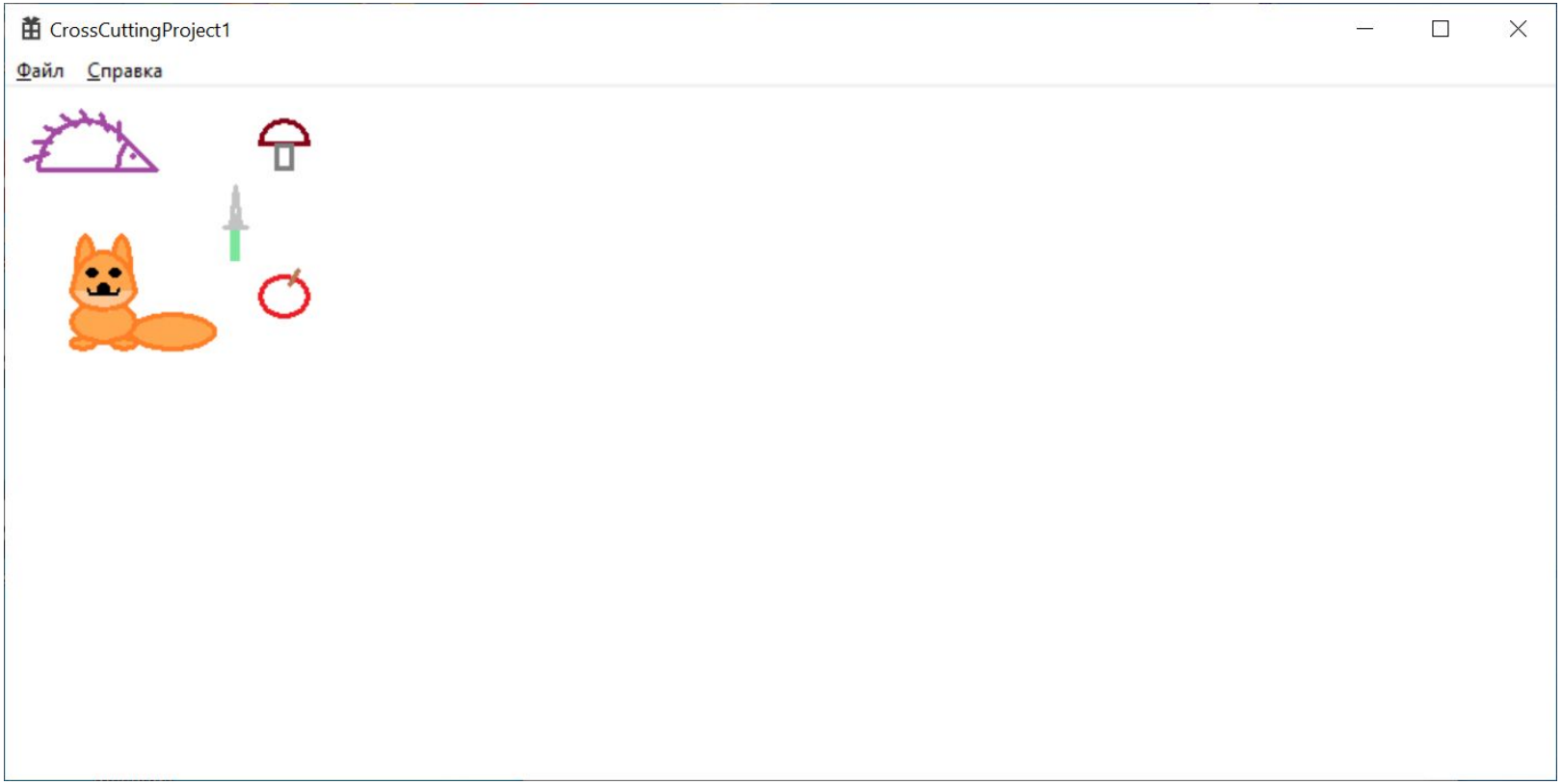
До утверждения темы не рекомендуется начинать кодирование.

Сразу после утверждения вы можете выполнять Задачу 3 и далее.

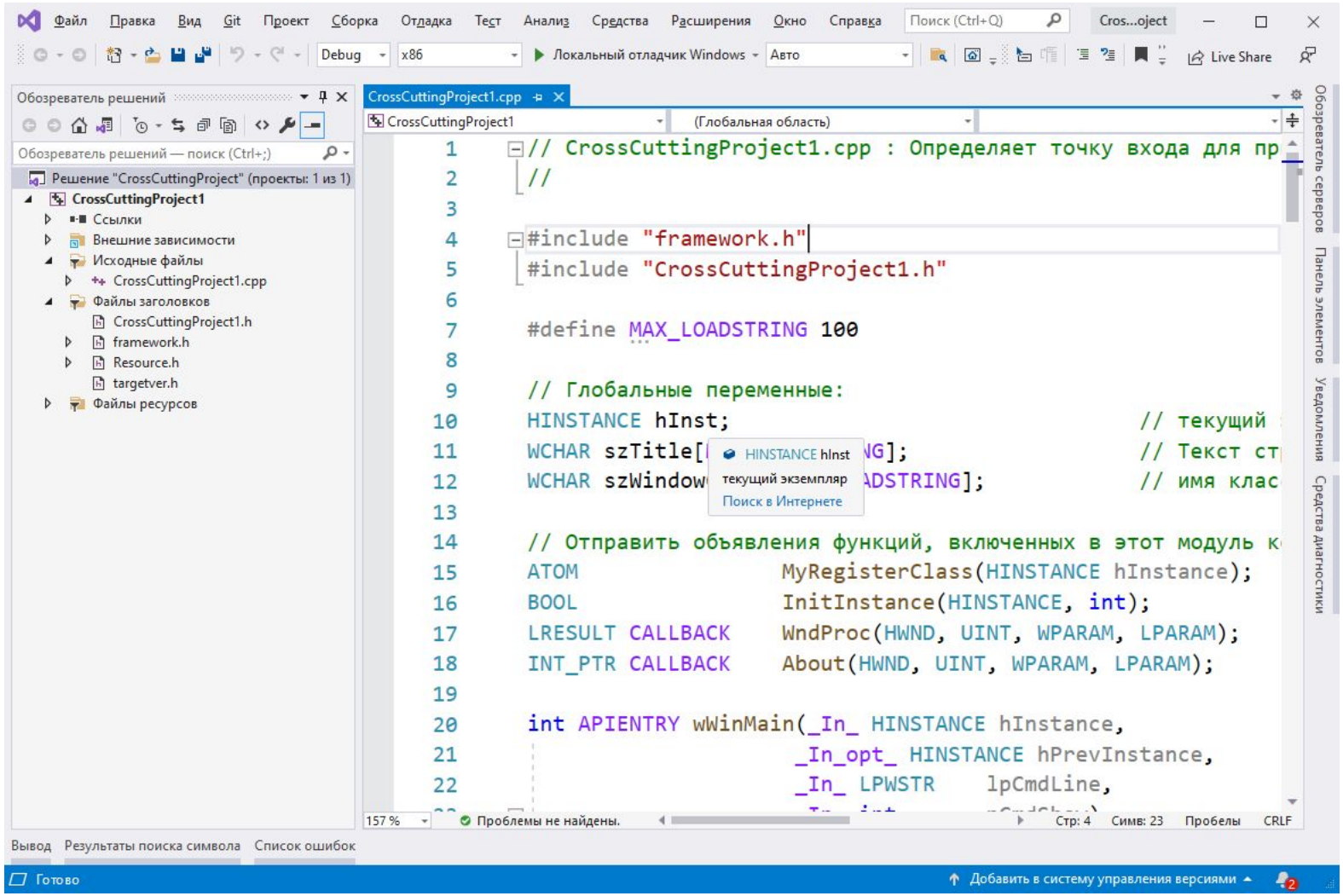
Задача 3. Рисуем все персонажи и предметы в виде картинки (в компьютере или на бумаге)



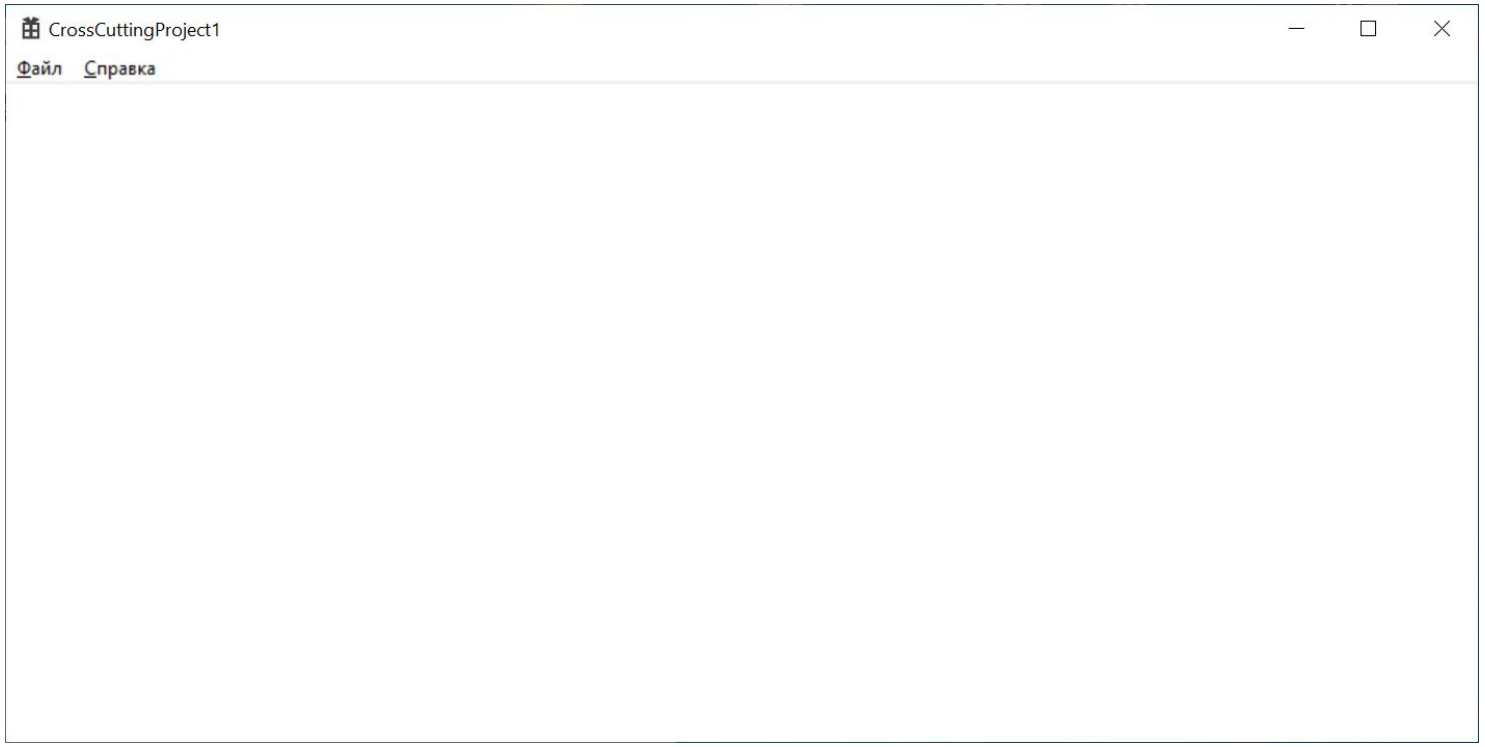
Задача 4. Создать приложение, содержащее все объекты из игры



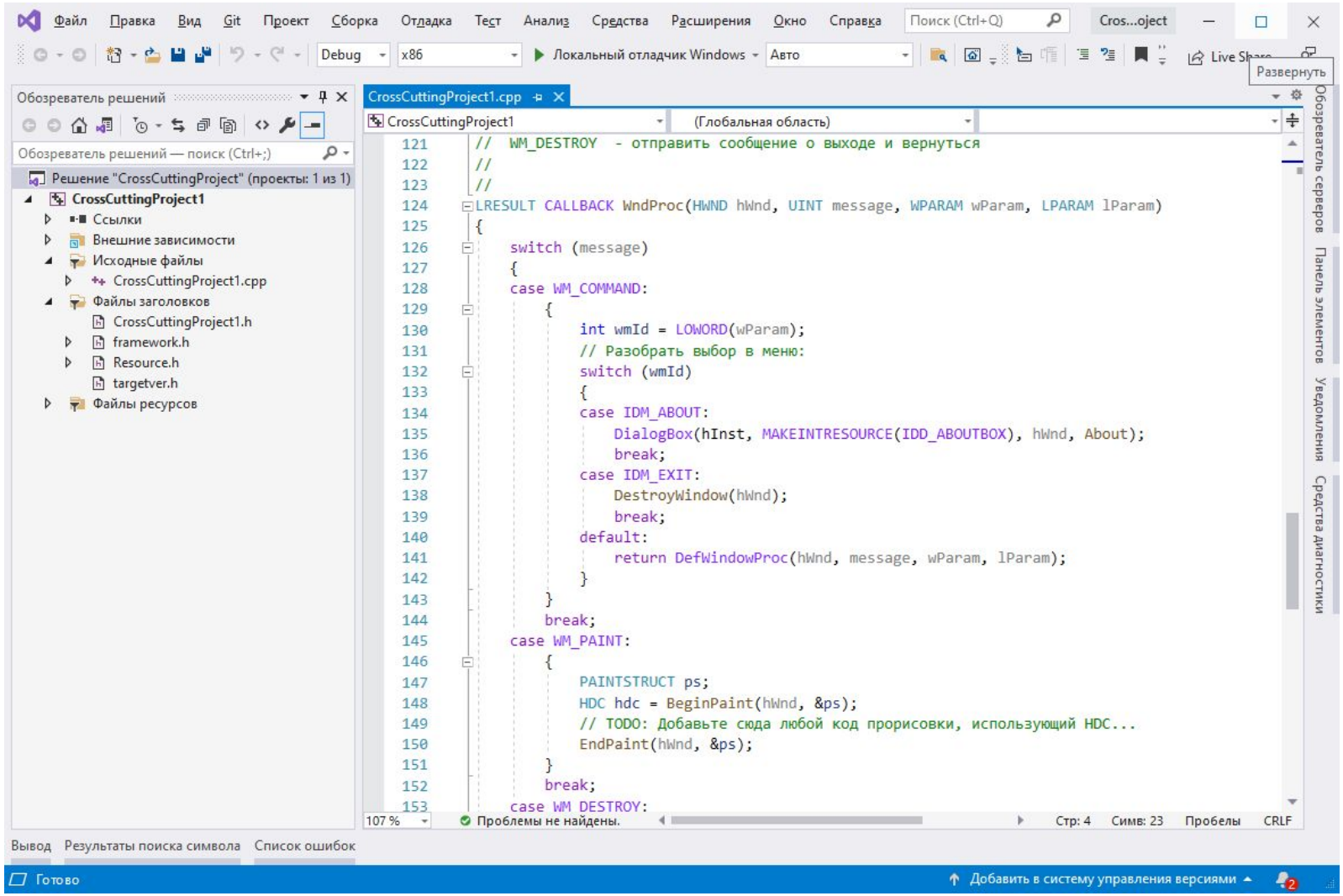
Шаг 1. Создаем проект



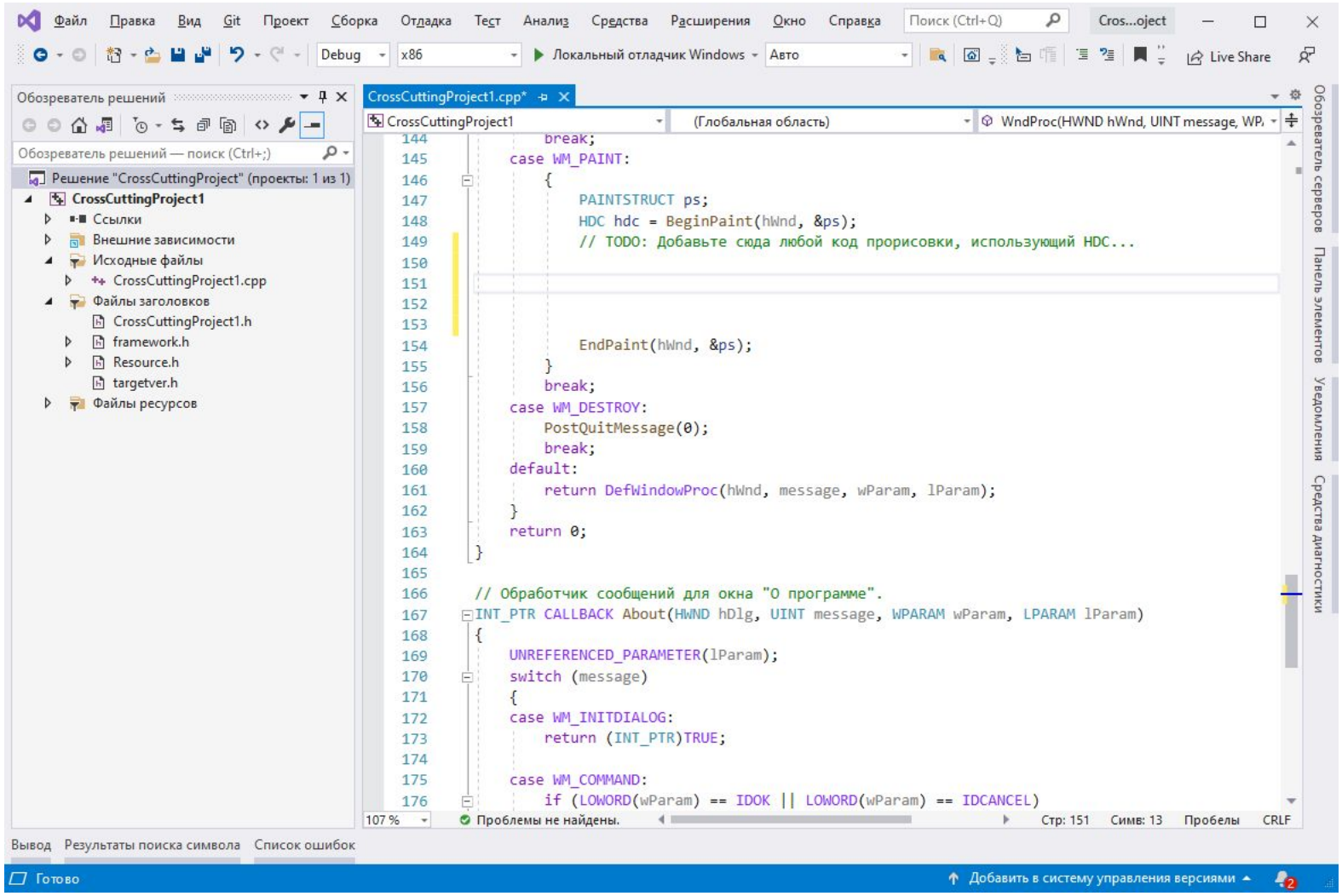
Шаг 2. Запускаем приложение



Шаг 3. Ищем WndProc



Шаг 4. Ищем “case WM_PAINT:”



Основной шаг: Добавляем код, рисующий все объекты! (1)

CrossCuttingProject1.cpp

CrossCuttingProject1 (Глобальная область)

```
148             break;
149         case WM_PAINT:
150             {
151                 PAINTSTRUCT ps;
152                 HDC hdc = BeginPaint(hWnd, &ps);
153                 // TODO: Добавьте сюда любой код прорисовки, использующий HDC...
154
155
156                 // Гриб
157                 // Шляпка гриба
158                 HPEN hPenMushroom1 = CreatePen(PS_SOLID, 3, RGB(136, 0, 21));
159                 SelectObject(hdc, hPenMushroom1);
160
161                 Chord(hdc, 155, 20, 185, 50, 185, 35, 155, 35);
162
163                 DeleteObject(hPenMushroom1);
164
165                 // Ножка гриба
166                 HPEN hPenMushroom2 = CreatePen(PS_SOLID, 3, RGB(127, 127, 127));
167                 SelectObject(hdc, hPenMushroom2);
168
169                 Rectangle(hdc, 165, 35, 175, 50);
170
171                 DeleteObject(hPenMushroom2);
172
173
174                 // Нож
175                 // Ручка ножа
176                 HPEN hPenKnife1 = CreatePen(PS_SOLID, 3, RGB(124, 231, 156));
177                 SelectObject(hdc, hPenKnife1);
178
179                 Rectangle(hdc, 138, 85, 142, 105);
180
181
182
183                 // Лезвие ножа
184                 HPEN hPenKnife2 = CreatePen(PS_SOLID, 3, RGB(195, 195, 195));
185                 SelectObject(hdc, hPenKnife2);
186
187                 MoveToEx(hdc, 137, 85, NULL);
188                 LineTo(hdc, 140, 60);
189                 LineTo(hdc, 143, 85);
190                 MoveToEx(hdc, 133, 85, NULL);
191                 LineTo(hdc, 147, 85);
192                 MoveToEx(hdc, 140, 78, NULL);
193                 LineTo(hdc, 140, 85);
194
```

CrossCuttingProject1.cpp*

CrossCuttingProject1 (Глобальная область)

```
194
195                 DeleteObject(hPenKnife1);
196                 DeleteObject(hPenKnife2);
197
198
199                 // Яблоко
200                 // Само яблоко
201                 HPEN hPenApple1 = CreatePen(PS_SOLID, 3, RGB(237, 28, 36));
202                 SelectObject(hdc, hPenApple1);
203
204                 Ellipse(hdc, 155, 115, 185, 140);
205
206                 DeleteObject(hPenApple1);
207
208                 // Хвостик яблока
209                 HPEN hPenApple2 = CreatePen(PS_SOLID, 3, RGB(185, 122, 87));
210                 SelectObject(hdc, hPenApple2);
211
212                 MoveToEx(hdc, 173, 120, NULL);
213                 LineTo(hdc, 178, 111);
214
215                 DeleteObject(hPenApple2);
216
217
218                 // Ёжик
219                 HPEN hPenHedgehog1 = CreatePen(PS_SOLID, 3, RGB(163, 73, 164));
220                 SelectObject(hdc, hPenHedgehog1);
221
222                 // Спина
223                 Arc(hdc, 20, 20, 80, 80, 80, 25, 20, 50);
224
225                 // Мордочка и живот
226                 MoveToEx(hdc, 72, 31, NULL);
227                 LineTo(hdc, 92, 50);
228                 LineTo(hdc, 20, 50);
229                 Arc(hdc, 68, 24, 118, 76, 72, 31, 0, 50);
230
231                 // Глаз
232                 Ellipse(hdc, 77, 41, 79, 43);
233
234                 // Иголки
235                 MoveToEx(hdc, 26, 40, NULL);
236                 LineTo(hdc, 12, 44);
237
238                 MoveToEx(hdc, 28, 33, NULL);
239                 LineTo(hdc, 17, 33);
240
```

Основной шаг: Добавляем код, рисующий все объекты! (2)

CrossCuttingProject1.cpp*

CrossCuttingProject1 (Глобальная область)

```
239             LineTo(hdc, 17, 33);
240
241             MoveToEx(hdc, 33, 27, NULL);
242             LineTo(hdc, 24, 24);
243
244             MoveToEx(hdc, 43, 23, NULL);
245             LineTo(hdc, 34, 17);
246
247             MoveToEx(hdc, 51, 22, NULL);
248             LineTo(hdc, 46, 14);
249
250             MoveToEx(hdc, 62, 26, NULL);
251             LineTo(hdc, 58, 16);
252
253             MoveToEx(hdc, 69, 32, NULL);
254             LineTo(hdc, 69, 21);
255
256
257             DeleteObject(hPenHedgehog1);
258
259
260             // Лиса
261             HPEN hPenFox1 = CreatePen(PS_SOLID, 3, RGB(255, 127, 39));
262             HPEN hPenFox2 = CreatePen(PS_SOLID, 3, RGB(255, 200, 150));
263             HPEN hPenFox3 = CreatePen(PS_SOLID, 3, RGB(0, 0, 0));
264             HPEN hPenFox4 = CreatePen(PS_SOLID, 2, RGB(0, 0, 0));
265
266             HBRUSH hBrushFox1 = CreateSolidBrush(RGB(255, 167, 79));
267             HBRUSH hBrushFox2 = CreateSolidBrush(RGB(255, 200, 150));
268
269             SelectObject(hdc, hPenFox1);
270             SelectObject(hdc, hBrushFox1);
271
272             // Хвост
273             Ellipse(hdc, 76, 138, 128, 160);
274
275             // Туловище
276             Ellipse(hdc, 40, 130, 80, 156);
277
278             // Лапы
279             Ellipse(hdc, 40, 152, 55, 160);
280             Ellipse(hdc, 65, 152, 80, 160);
281
282
283
284
285             // Уши
```

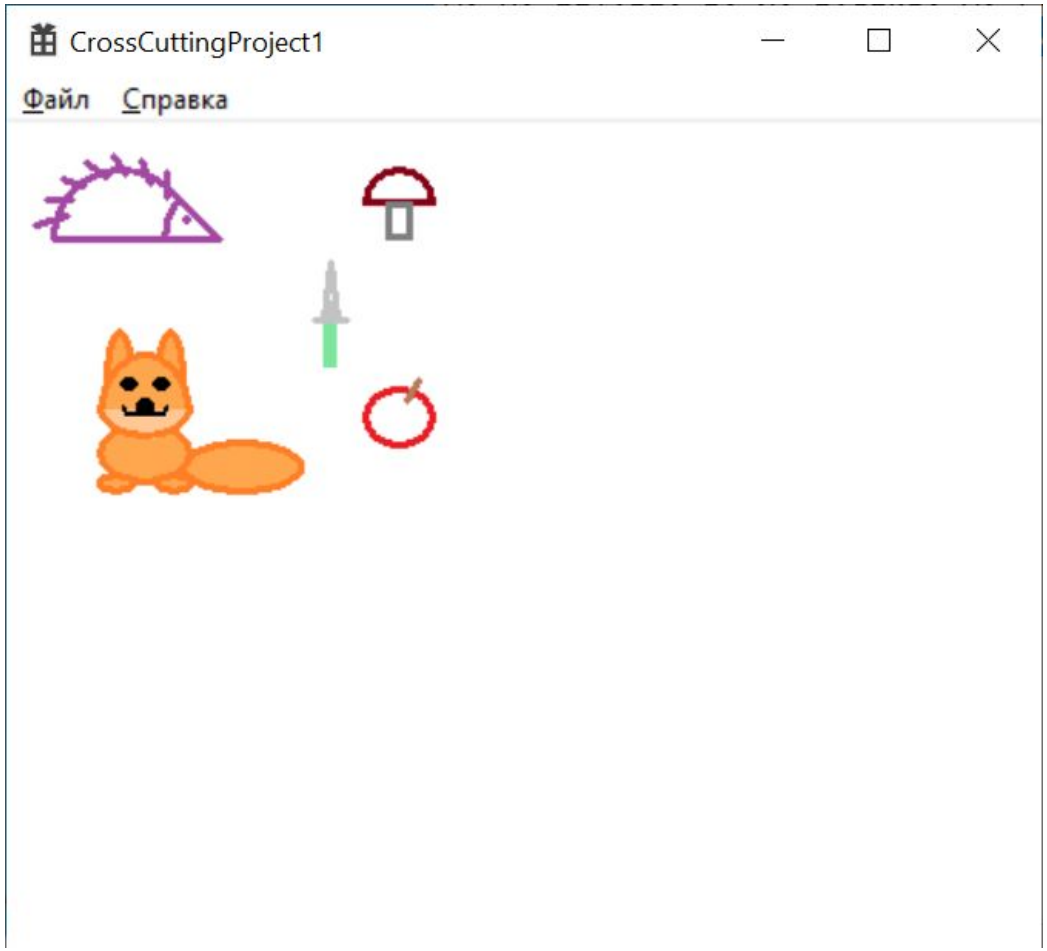
CrossCuttingProject1.cpp*

CrossCuttingProject1 (Глобальная область)

```
285             // Уши
286             SelectObject(hdc, hPenFox1);
287             SelectObject(hdc, hBrushFox1);
288             Chord(hdc, 43, 90, 55, 134, 100, 120, 20, 120);
289             Chord(hdc, 65, 90, 77, 134, 100, 120, 20, 120);
290
291             // Нижняя часть мордочки
292             SelectObject(hdc, hPenFox2);
293             SelectObject(hdc, hBrushFox2);
294             Chord(hdc, 40, 110, 80, 136, 20, 123, 100, 123);
295
296             SelectObject(hdc, hBrushFox1);
297             // Верхняя часть мордочки
298             Chord(hdc, 40, 100, 80, 155, 100, 123, 20, 123);
299
300             // контур мордочки
301             SelectObject(hdc, hPenFox1);
302             // Нижняя часть
303             Arc(hdc, 40, 110, 80, 136, 20, 123, 100, 123);
304
305             // верхняя часть
306             Arc(hdc, 40, 100, 80, 155, 100, 123, 20, 123);
307
308             // Глаза
309             SelectObject(hdc, hPenFox3);
310             Ellipse(hdc, 53-3, 113-2, 53+3, 113+2);
311             Ellipse(hdc, 67-3, 113-2, 67+3, 113+2);
312
313             // Нос
314             Ellipse(hdc, 60 - 3, 123 - 3, 60 + 3, 123 + 3);
315
316             // рот
317             SelectObject(hdc, hPenFox4);
318             Arc(hdc, 60 - 10, 123 - 4, 60 , 123 +4, 20, 126, 100, 126);
319             Arc(hdc, 60, 123 - 4, 60 + 10, 123 + 4, 20, 126, 100, 126);
320
321             DeleteObject(hPenFox1);
322             DeleteObject(hPenFox2);
323             DeleteObject(hPenFox3);
324             DeleteObject(hPenFox4);
325             DeleteObject(hBrushFox1);
326             DeleteObject(hBrushFox2);
327
328             EndPaint(hWnd, &ps);
329         }
330         break;
331     case WM_DESTROY:
```

107 % Проблемы не найдены.

Получившееся приложение со всеми объектами игры



Задача 5. Презентация преподавателю получившегося приложения на занятии

Вам необходимо показывать ваши наработки по игре преподавателю после выполнения КАЖДОГО ЭТАПА проекта!

Если вы выполнили все этапы, но показали всё только в конце, то проект вам не засчитывается. ДАЖЕ ЕСЛИ ВЫ ДОКАЗАЛИ полное свое авторство – увы ☹️

ИТОГО по этапу 1

1. Вы выбрали вариант игры и согласовали его с преподавателем
2. Описали игру – объекты и сценарии
3. Создали картинку (на бумаге или в компьютере) с объектами
4. Создали программу отрисовывающую все объекты
5. Продемонстрировали программу и картинку преподавателю

У вас есть всё, чтобы идти дальше и реализовать сценарии игры!

Сквозной проект – этап 2 (СП2)

Объекты игры распределяются по отдельным функциям.
Добавляется управление героем.
Игра заканчивается при победе.
Презентация итогов этапа 2 преподавателю.

Задачи ЭТАПА 2

Задача 1. Для каждого объекта создать свою собственную функцию
Задача 2. Управление героем через клавиатуру
Задача 3. Игра автоматически заканчивается при победе (ёжик нашел гриб)

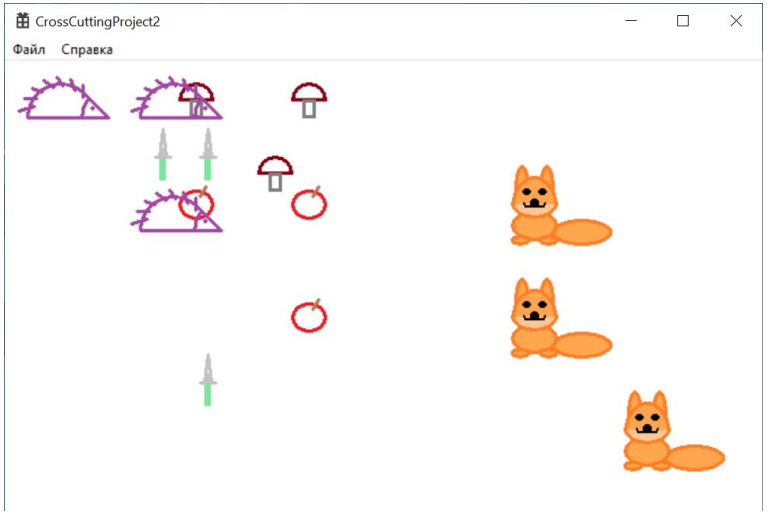
Задача 4. Презентация преподавателю получившегося приложения на занятии

Задача 1. Для каждого объекта создать свою собственную функцию

Для каждого объекта нужно создать функцию вида

```
void DrawMushroom(HDC hdc, int cx, int cy) {
    ...
}
void DrawKnife(HDC hdc, int cx, int cy) ...
void DrawApple(HDC hdc, int cx, int cy) ...
void DrawHedgehog(HDC hdc, int cx, int cy) ...
void DrawFox(HDC hdc, int cx, int cy) ...
```

в которые нужно перенести весь код отрисовки соответствующих объектов. Сделать тестовую отрисовку 3 объектов каждого типа



Задача 1. Реализация в игре «Жизнь ёжика» (1)

CrossCuttingProject2.cpp* | CrossCuttingProject1.cpp

CrossCuttingProject2 | (Глобальная область) | InitInstance(HI

```
95  //
96  BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
97  {
98     hInst = hInstance; // Сохранить маркер экземпляра в глобальной переменной
99
100    HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
101       CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);
102
103    if (!hWnd)
104    {
105       return FALSE;
106    }
107
108    ShowWindow(hWnd, nCmdShow);
109    UpdateWindow(hWnd);
110
111    return TRUE;
112 }
113
114
115 void DrawMushroom(HDC hdc, int cx, int cy) {
116     // dx = -170,  dy = -35
117
118     // Гриб
119     // Шляпка гриба
120     HPEN hPenMushroom1 = CreatePen(PS_SOLID, 3, RGB(136, 0, 21));
121     SelectObject(hdc, hPenMushroom1);
122
123     Chord(hdc, 155 - 170 + cx , 20 - 35 + cy, 185 - 170 + cx, 50 - 35 + cy, 185 - 170 + cx, 35 - 35 + cy, 155 - 170 + cx, 35 - 35 + cy);
124
125     DeleteObject(hPenMushroom1);
126
127     // Ножка гриба
128     HPEN hPenMushroom2 = CreatePen(PS_SOLID, 3, RGB(127, 127, 127));
129     SelectObject(hdc, hPenMushroom2);
130
131     Rectangle(hdc, 165 - 170 + cx, 35 - 35 + cy, 175 - 170 + cx, 50 - 35 + cy);
132
133     DeleteObject(hPenMushroom2);
134 }
135
136
137 void DrawKnife(HDC hdc, int cx, int cy) {
138     // dx = -140,  dy = -85
139
140     // Нож
141     // Ручка ножа
```

107 % | Проблемы не найдены.

Задача 1. Реализация в игре «Жизнь ёжика» (2)

CrossCuttingProject2.cpp* | CrossCuttingProject1.cpp

CrossCuttingProject2 | (Глобальная область)

```
137 void DrawKnife(HDC hdc, int cx, int cy) {
138     // dx = -140,  dy = -85
139
140     // Нож
141     // Ручка ножа
142     HPEN hPenKnife1 = CreatePen(PS_SOLID, 3, RGB(124, 231, 156));
143     SelectObject(hdc, hPenKnife1);
144
145     Rectangle(hdc, 138 - 140 + cx, 85 - 85 + cy, 142 - 140 + cx, 105 - 85 + cy);
146
147     // Лезвие ножа
148     HPEN hPenKnife2 = CreatePen(PS_SOLID, 3, RGB(195, 195, 195));
149     SelectObject(hdc, hPenKnife2);
150
151     MoveToEx(hdc, 137 - 140 + cx, 85 - 85 + cy, NULL);
152     LineTo(hdc, 140 - 140 + cx, 60 - 85 + cy);
153     LineTo(hdc, 143 - 140 + cx, 85 - 85 + cy);
154     MoveToEx(hdc, 133 - 140 + cx, 85 - 85 + cy, NULL);
155     LineTo(hdc, 147 - 140 + cx, 85 - 85 + cy);
156     MoveToEx(hdc, 140 - 140 + cx, 78 - 85 + cy, NULL);
157     LineTo(hdc, 140 - 140 + cx, 85 - 85 + cy);
158
159     DeleteObject(hPenKnife1);
160     DeleteObject(hPenKnife2);
161 }
162
163
164 void DrawApple(HDC hdc, int cx, int cy) {
165     // dx = -170,  dy = -128
166
167     // Яблоко
168     // Само яблоко
169     HPEN hPenApple1 = CreatePen(PS_SOLID, 3, RGB(237, 28, 36));
170     SelectObject(hdc, hPenApple1);
171
172     Ellipse(hdc, 155 - 170 + cx, 115 - 128 + cy, 185 - 170 + cx, 140 - 128 + cy);
173     DeleteObject(hPenApple1);
174
175     // Хвостик яблока
176     HPEN hPenApple2 = CreatePen(PS_SOLID, 3, RGB(185, 122, 87));
177     SelectObject(hdc, hPenApple2);
178
179     MoveToEx(hdc, 173 - 170 + cx, 120 - 128 + cy, NULL);
180     LineTo(hdc, 178 - 170 + cx, 111 - 128 + cy);
181
182     DeleteObject(hPenApple2);
183 }
```

107 % | Проблемы не найдены.

Задача 1. Реализация в игре «Жизнь ёжика» (3)

CrossCuttingProject2.cpp* | CrossCuttingProject1.cpp

CrossCuttingProject2 | (Глобальная область)

```
184
185
186 void DrawHedgehog(HDC hdc, int cx, int cy) {
187     // dx = -56,  dy = -35
188
189     // Ёжик
190     HPEN hPenHedgehog1 = CreatePen(PS_SOLID, 3, RGB(163, 73, 164));
191     SelectObject(hdc, hPenHedgehog1);
192
193     // Спина
194     Arc(hdc, 20 - 56 + cx, 20 - 35 + cy, 80 - 56 + cx, 80 - 35 + cy, 80 - 56 + cx, 25 - 35 + cy, 20 - 56 + cx, 50 - 35 + cy);
195
196     // Мордочка и живот
197     MoveToEx(hdc, 72 - 56 + cx, 31 - 35 + cy, NULL);
198     LineTo(hdc, 92 - 56 + cx, 50 - 35 + cy);
199     LineTo(hdc, 20 - 56 + cx, 50 - 35 + cy);
200     Arc(hdc, 68 - 56 + cx, 24 - 35 + cy, 118 - 56 + cx, 76 - 35 + cy, 72 - 56 + cx, 31 - 35 + cy, 0 - 56 + cx, 50 - 35 + cy);
201
202     // Глаз
203     Ellipse(hdc, 77 - 56 + cx, 41 - 35 + cy, 79 - 56 + cx, 43 - 35 + cy);
204
205     // Иголки
206     MoveToEx(hdc, 26 - 56 + cx, 40 - 35 + cy, NULL);
207     LineTo(hdc, 12 - 56 + cx, 44 - 35 + cy);
208
209     MoveToEx(hdc, 28 - 56 + cx, 33 - 35 + cy, NULL);
210     LineTo(hdc, 17 - 56 + cx, 33 - 35 + cy);
211
212     MoveToEx(hdc, 33 - 56 + cx, 27 - 35 + cy, NULL);
213     LineTo(hdc, 24 - 56 + cx, 24 - 35 + cy);
214
215     MoveToEx(hdc, 43 - 56 + cx, 23 - 35 + cy, NULL);
216     LineTo(hdc, 34 - 56 + cx, 17 - 35 + cy);
217
218     MoveToEx(hdc, 51 - 56 + cx, 22 - 35 + cy, NULL);
219     LineTo(hdc, 46 - 56 + cx, 14 - 35 + cy);
220
221     MoveToEx(hdc, 62 - 56 + cx, 26 - 35 + cy, NULL);
222     LineTo(hdc, 58 - 56 + cx, 16 - 35 + cy);
223
224     MoveToEx(hdc, 69 - 56 + cx, 32 - 35 + cy, NULL);
225     LineTo(hdc, 69 - 56 + cx, 21 - 35 + cy);
226
227
228     DeleteObject(hPenHedgehog1);
229 }
230
```

107 % | Проблемы не найдены.

Задача 1. Реализация в игре «Жизнь ёжика» (4)

CrossCuttingProject2.cpp* | CrossCuttingProject1.cpp

CrossCuttingProject2 | (Глобальная область)

```
229 }
230
231
232 void DrawFox(HDC hdc, int cx, int cy) {
233     // dx = -60,  dy = -125
234
235     // Лиса
236     HPEN hPenFox1 = CreatePen(PS_SOLID, 3, RGB(255, 127, 39));
237     HPEN hPenFox2 = CreatePen(PS_SOLID, 3, RGB(255, 200, 150));
238     HPEN hPenFox3 = CreatePen(PS_SOLID, 3, RGB(0, 0, 0));
239     HPEN hPenFox4 = CreatePen(PS_SOLID, 2, RGB(0, 0, 0));
240
241     HBRUSH hBrushFox1 = CreateSolidBrush(RGB(255, 167, 79));
242     HBRUSH hBrushFox2 = CreateSolidBrush(RGB(255, 200, 150));
243
244     SelectObject(hdc, hPenFox1);
245     SelectObject(hdc, hBrushFox1);
246
247     // Хвост
248     Ellipse(hdc, 76 - 60 + cx, 138 - 125 + cy, 128 - 60 + cx, 160 - 125 + cy);
249
250     // Туловище
251     Ellipse(hdc, 40 - 60 + cx, 130 - 125 + cy, 80 - 60 + cx, 156 - 125 + cy);
252
253     // Лапы
254     Ellipse(hdc, 40 - 60 + cx, 152 - 125 + cy, 55 - 60 + cx, 160 - 125 + cy);
255     Ellipse(hdc, 65 - 60 + cx, 152 - 125 + cy, 80 - 60 + cx, 160 - 125 + cy);
256
257     // Уши
258     SelectObject(hdc, hPenFox1);
259     SelectObject(hdc, hBrushFox1);
260     Chord(hdc, 43 - 60 + cx, 90 - 125 + cy, 55 - 60 + cx, 134 - 125 + cy, 100 - 60 + cx, 120 - 125 + cy, 20 - 60 + cx, 120 - 125 + cy);
261     Chord(hdc, 65 - 60 + cx, 90 - 125 + cy, 77 - 60 + cx, 134 - 125 + cy, 100 - 60 + cx, 120 - 125 + cy, 20 - 60 + cx, 120 - 125 + cy);
262
263     // Нижняя часть мордочки
264     SelectObject(hdc, hPenFox2);
265     SelectObject(hdc, hBrushFox2);
266     Chord(hdc, 40 - 60 + cx, 110 - 125 + cy, 80 - 60 + cx, 136 - 125 + cy, 20 - 60 + cx, 123 - 125 + cy, 100 - 60 + cx, 123 - 125 + cy);
267
268     SelectObject(hdc, hBrushFox1);
269     // Верхняя часть мордочки
270     Chord(hdc, 40 - 60 + cx, 100 - 125 + cy, 80 - 60 + cx, 155 - 125 + cy, 100 - 60 + cx, 123 - 125 + cy, 20 - 60 + cx, 123 - 125 + cy);
271
272     // контур мордочки
273     SelectObject(hdc, hPenFox1);
274     // Нижняя часть
275     Arc(hdc, 40 - 60 + cx, 110 - 125 + cy, 80 - 60 + cx, 136 - 125 + cy, 20 - 60 + cx, 123 - 125 + cy, 100 - 60 + cx, 123 - 125 + cy);
```

Задача 1. Реализация в игре «Жизнь ёжика» (5)

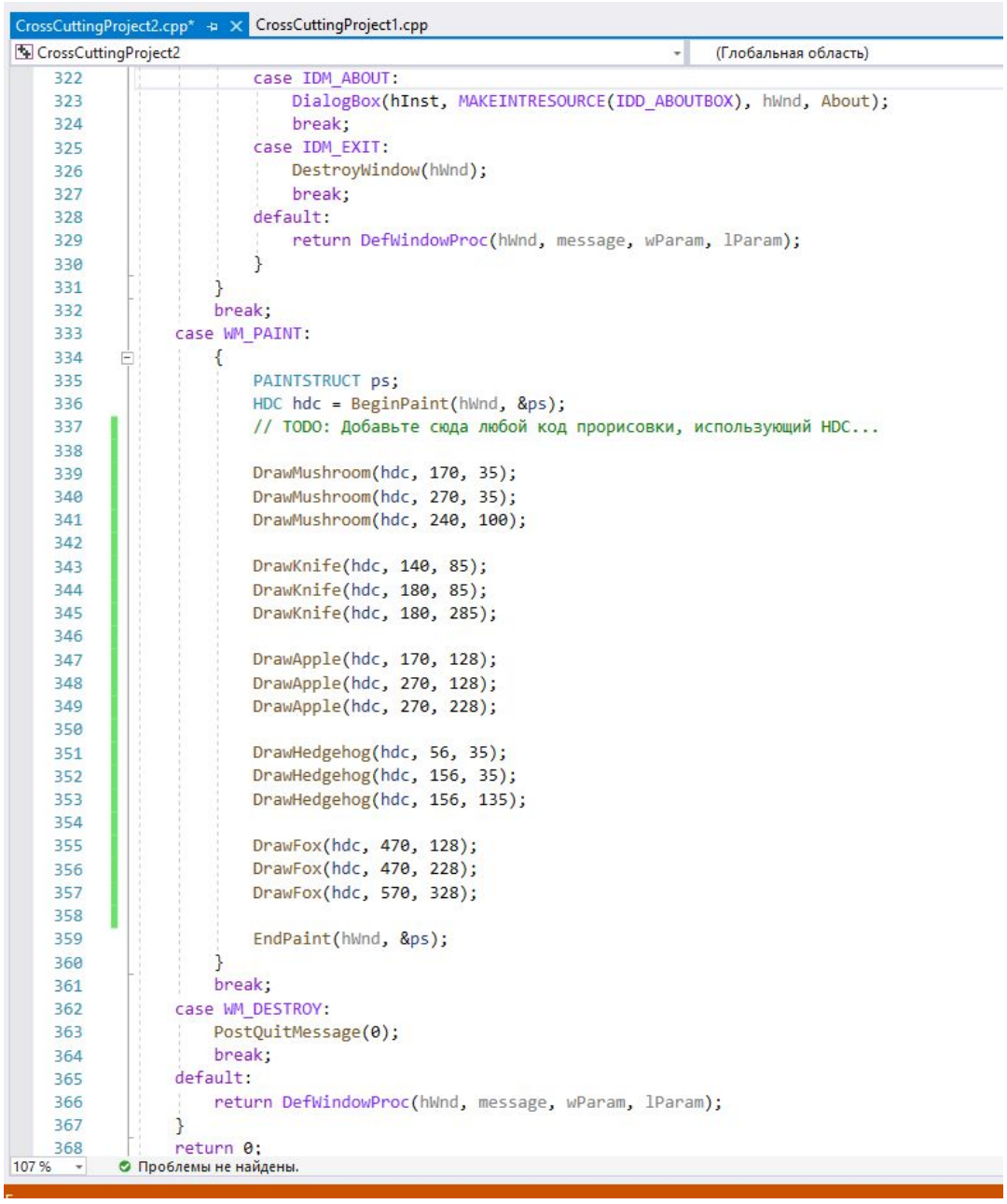
CrossCuttingProject2.cpp* | CrossCuttingProject1.cpp

CrossCuttingProject2 | (Глобальная область) | DrawFox(HDC hdc, int cx, int cy)

```
276
277     // верхняя часть
278     Arc(hdc, 40 - 60 + cx, 100 - 125 + cy, 80 - 60 + cx, 155 - 125 + cy, 100 - 60 + cx, 123 - 125 + cy, 20 - 60 + cx, 123 - 125 + cy);
279
280     // Глаза
281     SelectObject(hdc, hPenFox3);
282     Ellipse(hdc, 53 - 3 - 60 + cx, 113 - 2 - 125 + cy, 53 + 3 - 60 + cx, 113 + 2 - 125 + cy);
283     Ellipse(hdc, 67 - 3 - 60 + cx, 113 - 2 - 125 + cy, 67 + 3 - 60 + cx, 113 + 2 - 125 + cy);
284
285     // Нос
286     Ellipse(hdc, 60 - 3 - 60 + cx, 123 - 3 - 125 + cy, 60 + 3 - 60 + cx, 123 + 3 - 125 + cy);
287
288     // рот
289     SelectObject(hdc, hPenFox4);
290     Arc(hdc, 60 - 10 - 60 + cx, 123 - 4 - 125 + cy, 60 - 60 + cx, 123 + 4 - 125 + cy, 20 - 60 + cx, 126 - 125 + cy, 100 - 60 + cx, 126 - 125 + cy);
291     Arc(hdc, 60 - 60 + cx, 123 - 4 - 125 + cy, 60 + 10 - 60 + cx, 123 + 4 - 125 + cy, 20 - 60 + cx, 126 - 125 + cy, 100 - 60 + cx, 126 - 125 + cy);
292
293     DeleteObject(hPenFox1);
294     DeleteObject(hPenFox2);
295     DeleteObject(hPenFox3);
296     DeleteObject(hPenFox4);
297     DeleteObject(hBrushFox1);
298     DeleteObject(hBrushFox2);
299 }
300
301
302 //
303 //  ФУНКЦИЯ: WndProc(HWND, UINT, WPARAM, LPARAM)
304 //
305 //  ЦЕЛЬ: Обрабатывает сообщения в главном окне.
306 //
307 //  WM_COMMAND  - обработать меню приложения
308 //  WM_PAINT    - Отрисовка главного окна
309 //  WM_DESTROY  - отправить сообщение о выходе и вернуться
310 //
311 //
312 LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
313 {
314     switch (message)
315     {
316     case WM_COMMAND:
317         {
318             int wmId = LOWORD(wParam);
319             // Разобрать выбор в меню:
320             switch (wmId)
321             {
322             case IDM_ABOUT:
```

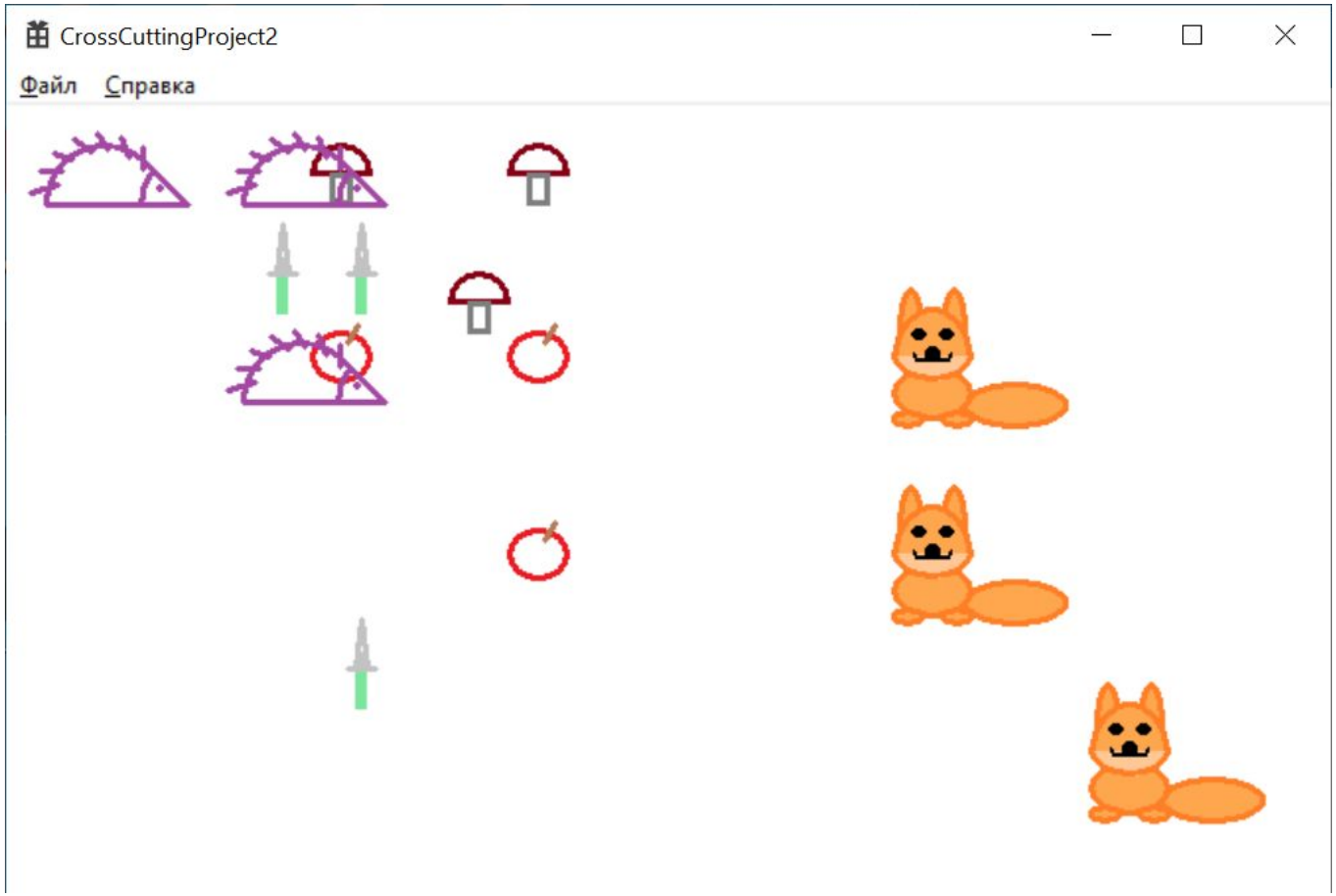
107 % | Проблемы не найдены.

Задача 1. Реализация в игре «Жизнь ёжика» (6)




```
322             case IDM_ABOUT:
323                 DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
324                 break;
325             case IDM_EXIT:
326                 DestroyWindow(hWnd);
327                 break;
328             default:
329                 return DefWindowProc(hWnd, message, wParam, lParam);
330             }
331         }
332         break;
333     case WM_PAINT:
334         {
335             PAINTSTRUCT ps;
336             HDC hdc = BeginPaint(hWnd, &ps);
337             // TODO: Добавьте сюда любой код прорисовки, использующий HDC...
338
339             DrawMushroom(hdc, 170, 35);
340             DrawMushroom(hdc, 270, 35);
341             DrawMushroom(hdc, 240, 100);
342
343             DrawKnife(hdc, 140, 85);
344             DrawKnife(hdc, 180, 85);
345             DrawKnife(hdc, 180, 285);
346
347             DrawApple(hdc, 170, 128);
348             DrawApple(hdc, 270, 128);
349             DrawApple(hdc, 270, 228);
350
351             DrawHedgehog(hdc, 56, 35);
352             DrawHedgehog(hdc, 156, 35);
353             DrawHedgehog(hdc, 156, 135);
354
355             DrawFox(hdc, 470, 128);
356             DrawFox(hdc, 470, 228);
357             DrawFox(hdc, 570, 328);
358
359             EndPaint(hWnd, &ps);
360         }
361         break;
362     case WM_DESTROY:
363         PostQuitMessage(0);
364         break;
365     default:
366         return DefWindowProc(hWnd, message, wParam, lParam);
367     }
368     return 0;
```

Задача 1. Работающее приложение



Задача 2. Управление героем через клавиатуру

Реализовать в программе управление героем через клавиатуру.

Для этого нужно:

1) Завести глобальные переменные, в которых хранятся координаты героя
2) В WM_PAINT при отрисовке героя подставлять значения этих глобальных переменных
3) Добавить обработку клавиш. При нажатии клавиши «ВЛЕВО» уменьшать X координату героя. При нажатии клавиши «ВПРАВО» увеличивать X координату героя. Аналогично изменять координату Y.

Задача 2. Реализация (1). Глобальные переменные

CrossCuttingProject_2_1.cpp* | CrossCuttingProject2.cpp | CrossCuttingProject1.cpp

CrossCuttingProject2_1 | (Глобальная область)

```
1   // CrossCuttingProject_2_1.cpp : Определяет точку входа для приложения.
2   //
3
4   #include "framework.h"
5   #include "CrossCuttingProject_2_1.h"
6
7   #define MAX_LOADSTRING 100
8
9   // Глобальные переменные:
10  HINSTANCE hInst;                                // текущий экземпляр
11  WCHAR szTitle[MAX_LOADSTRING];                  // Текст строки заголовка
12  WCHAR szWindowClass[MAX_LOADSTRING];            // имя класса главного окна
13
14  int HedgehogX = 100;
15  int HedgehogY = 200;
16
17  // Отправить объявления функций, включенных в этот модуль кода:
18  ATOM                MyRegisterClass(HINSTANCE hInstance);
19  BOOL                InitInstance(HINSTANCE, int);
20  LRESULT CALLBACK    WndProc(HWND, UINT, WPARAM, LPARAM);
21  INT_PTR CALLBACK    About(HWND, UINT, WPARAM, LPARAM);
22
23  int APIENTRY wWinMain(_In_ HINSTANCE hInstance,
24                        _In_opt_ HINSTANCE hPrevInstance,
25                        _In_ LPWSTR    lpCmdLine,
26                        _In_ int       nCmdShow)
27  {
28      UNREFERENCED_PARAMETER(hPrevInstance);
29      UNREFERENCED_PARAMETER(lpCmdLine);
```

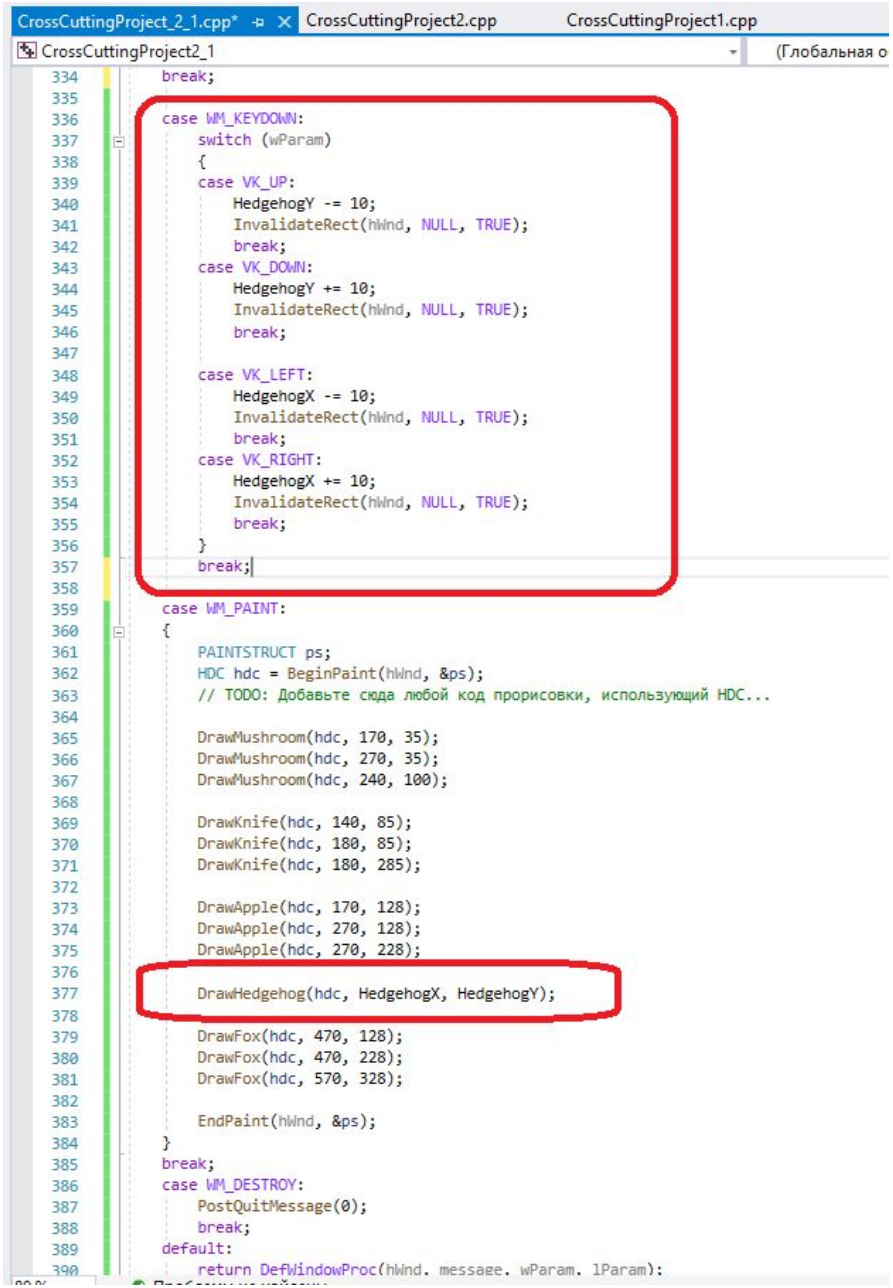
Задача 2. Реализация (2). Изменения в WndProc

CrossCuttingProject_2_1.cpp* | CrossCuttingProject2.cpp | CrossCuttingProject1.cpp

CrossCuttingProject2_1 | (Глобальная область)

```
301     }
302
303
304     //
305     //  ФУНКЦИЯ: WndProc(HWND, UINT, WPARAM, LPARAM)
306     //
307     //  ЦЕЛЬ: Обрабатывает сообщения в главном окне.
308     //
309     //  WM_COMMAND  - обработать меню приложения
310     //  WM_PAINT    - Отрисовка главного окна
311     //  WM_DESTROY  - отправить сообщение о выходе и вернуться
312     //
313     //
314     LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
315     {
316         switch (message)
317         {
318         case WM_COMMAND:
319         {
320             int wmId = LOWORD(wParam);
321             // Разобрать выбор в меню:
322             switch (wmId)
323             {
324             case IDM_ABOUT:
325                 DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
326                 break;
327             case IDM_EXIT:
328                 DestroyWindow(hWnd);
329                 break;
330             default:
331                 return DefWindowProc(hWnd, message, wParam, lParam);
332             }
333         }
334         break;
335
336         case WM_KEYDOWN:
337             switch (wParam)
338             {
339             case VK_UP:
340                 HedgehogY -= 10;
341                 InvalidateRect(hWnd, NULL, TRUE);
342                 break;
343             case VK_DOWN:
344                 HedgehogY += 10;
345                 InvalidateRect(hWnd, NULL, TRUE);
346                 break;
347
```

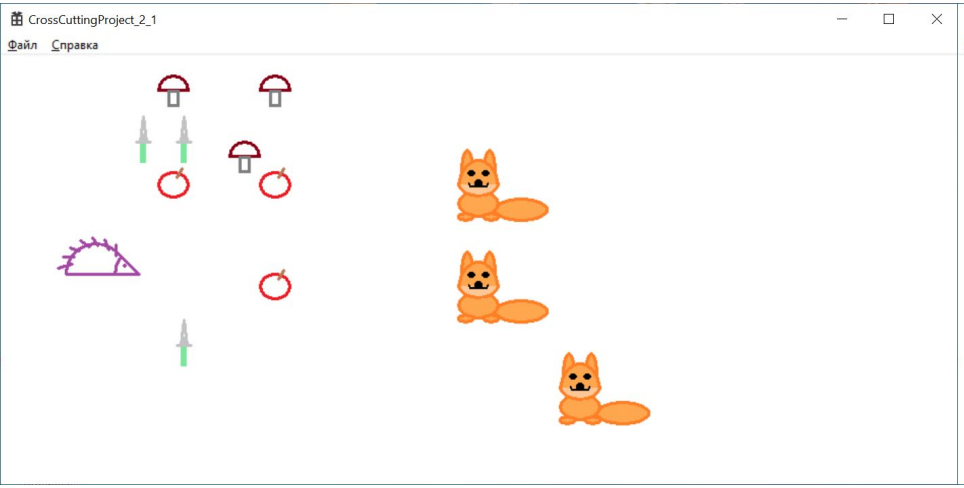
Задача 2. Реализация (3). Изменения в WndProc

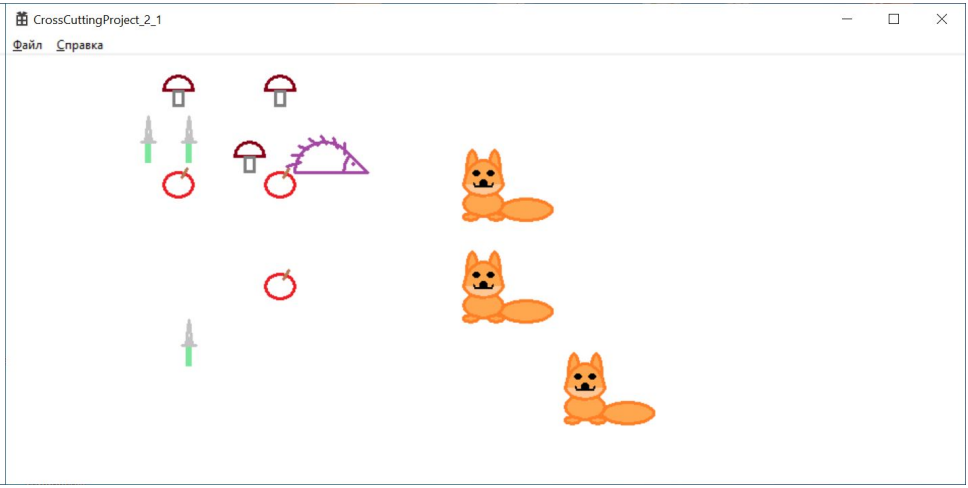


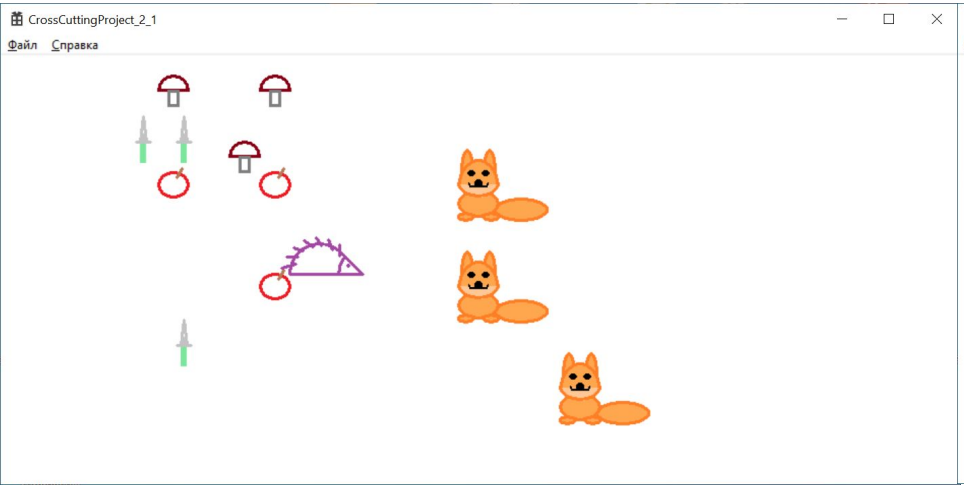
Задача 2. Код обработки нажатий клавиш

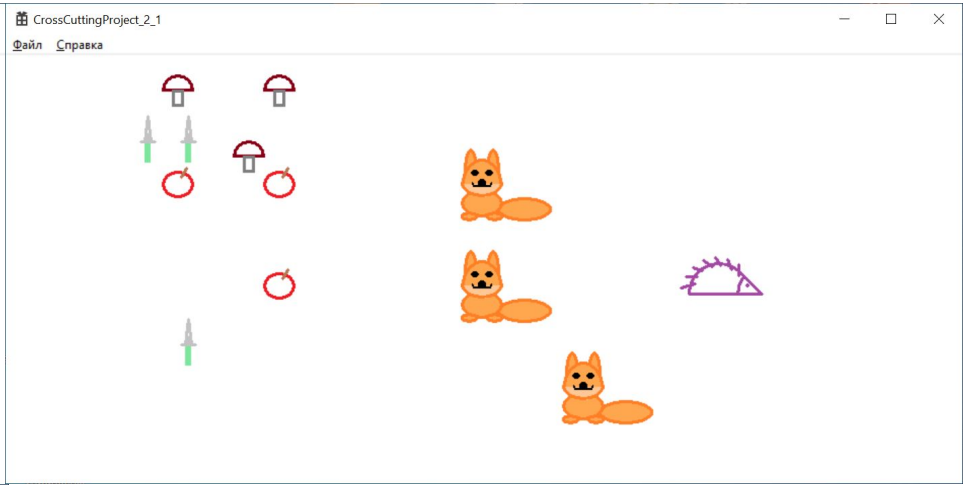
```
case WM_KEYDOWN:
        switch (wParam)
        {
        case VK_UP:
            HedgehogY -= 10;
            InvalidateRect(hWnd, NULL, TRUE);
            break;
        case VK_DOWN:
            HedgehogY += 10;
            InvalidateRect(hWnd, NULL, TRUE);
            break;

        case VK_LEFT:
            HedgehogX -= 10;
            InvalidateRect(hWnd, NULL, TRUE);
            break;
        case VK_RIGHT:
            HedgehogX += 10;
            InvalidateRect(hWnd, NULL, TRUE);
            break;
        }
        break;
```

Задача 2. Работающее приложение









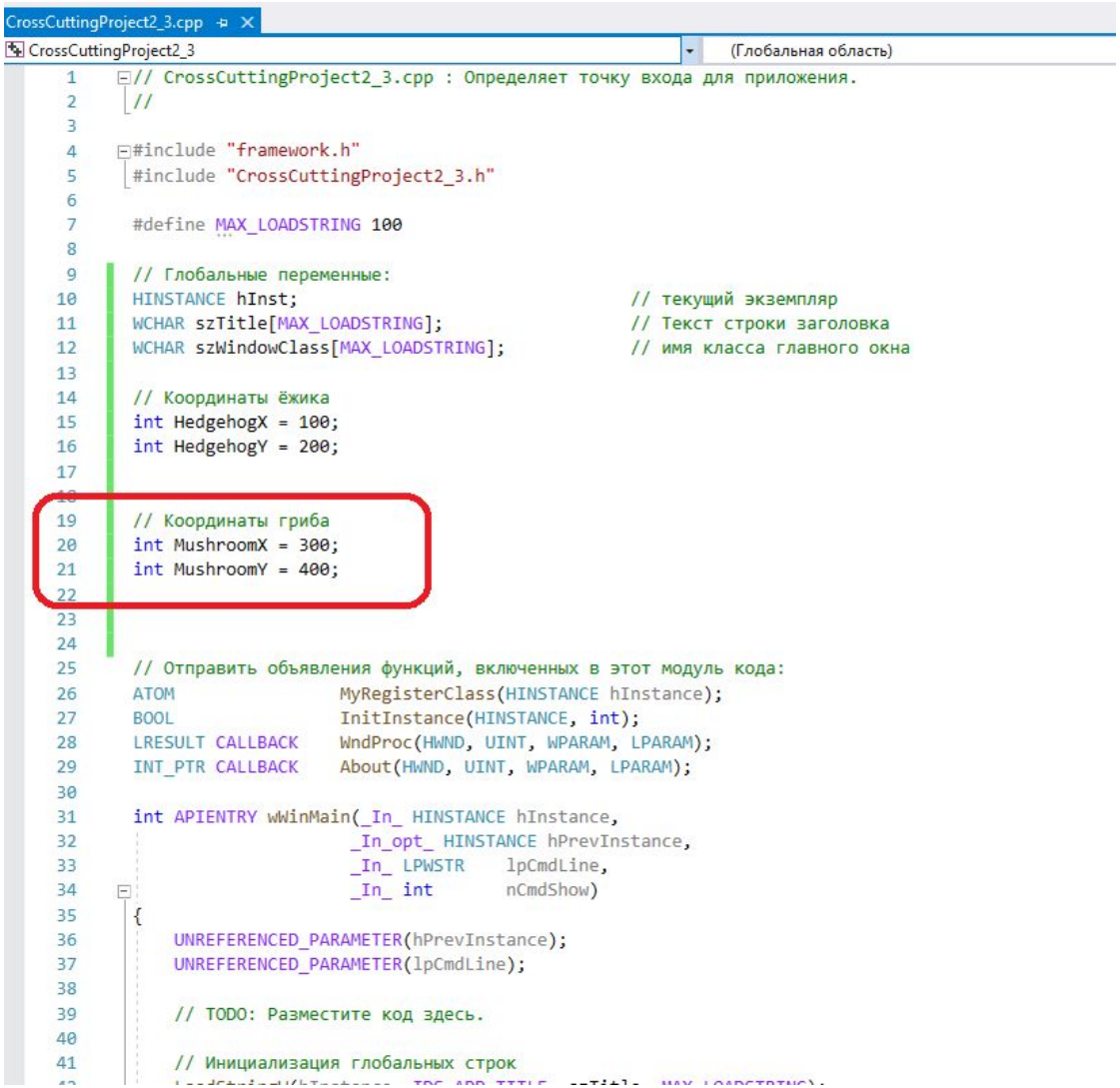
Задача 3. Игра автоматически заканчивается при победе

Реализовать в программе завершение программы если герой нашел необходимый ему предмет.

Для этого нужно:

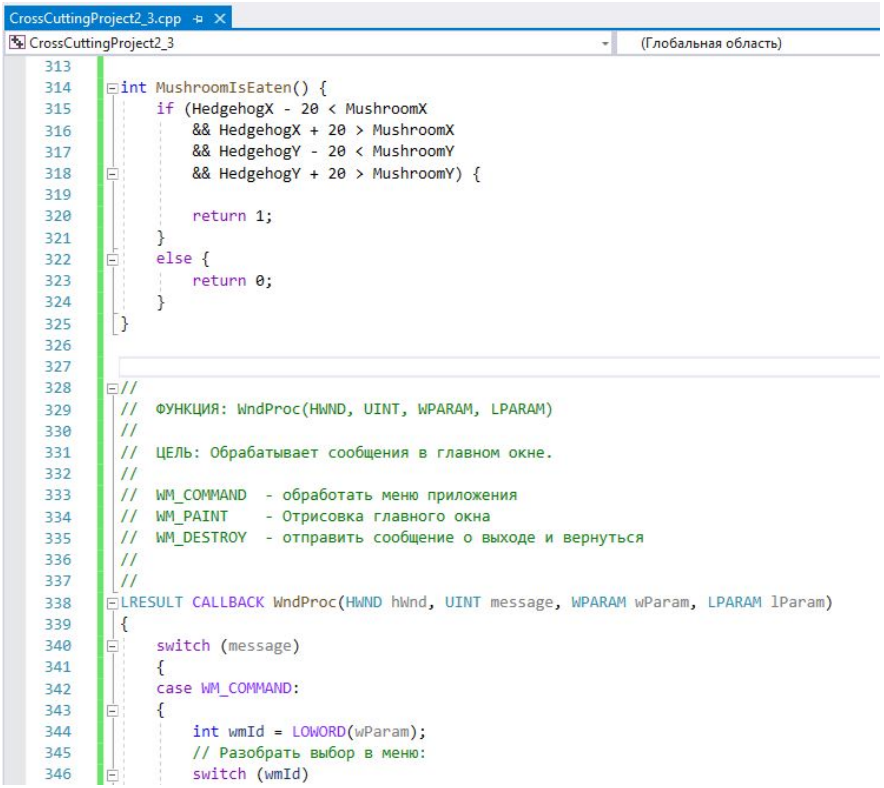
1) Завести глобальные переменные, в которых хранятся координаты нужного предмета
2) В WM_PAINT при отрисовке предмета подставлять значения этих глобальных переменных
3) В обработку клавиш добавить проверку совпадения координат героя и и координат предмета. В случае если эти координаты совпали – завершать игру

Задача 3. Реализация (1). Глобальные переменные

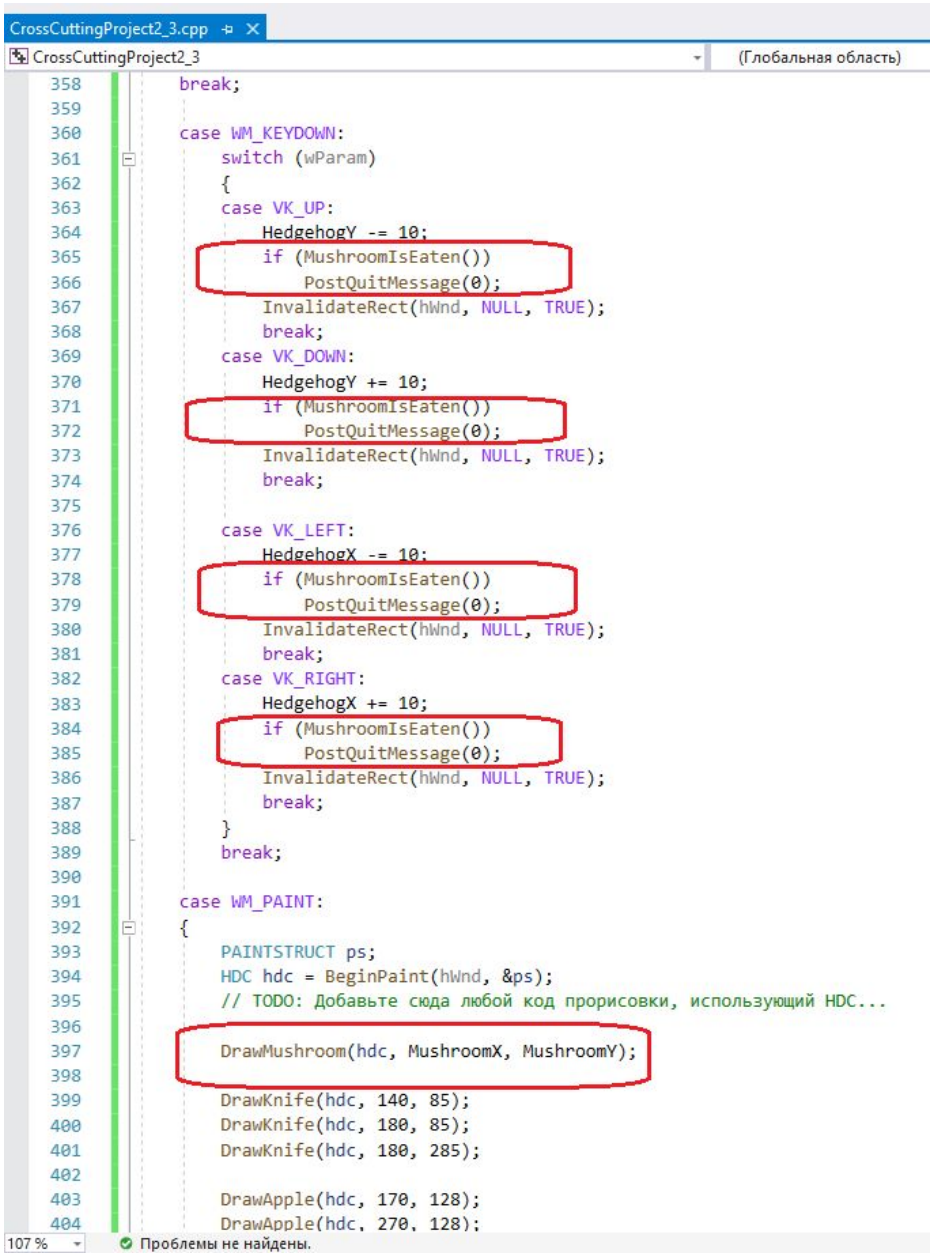


Задача 3. Реализация (2). Функция для проверки совпадения координат ёжика и гриба

```
int MushroomIsEaten() {
    if (HedgehogX - 20 < MushroomX
        && HedgehogX + 20 > MushroomX
        && HedgehogY - 20 < MushroomY
        && HedgehogY + 20 > MushroomY) {

        return 1;
    }
    else {
        return 0;
    }
}
```

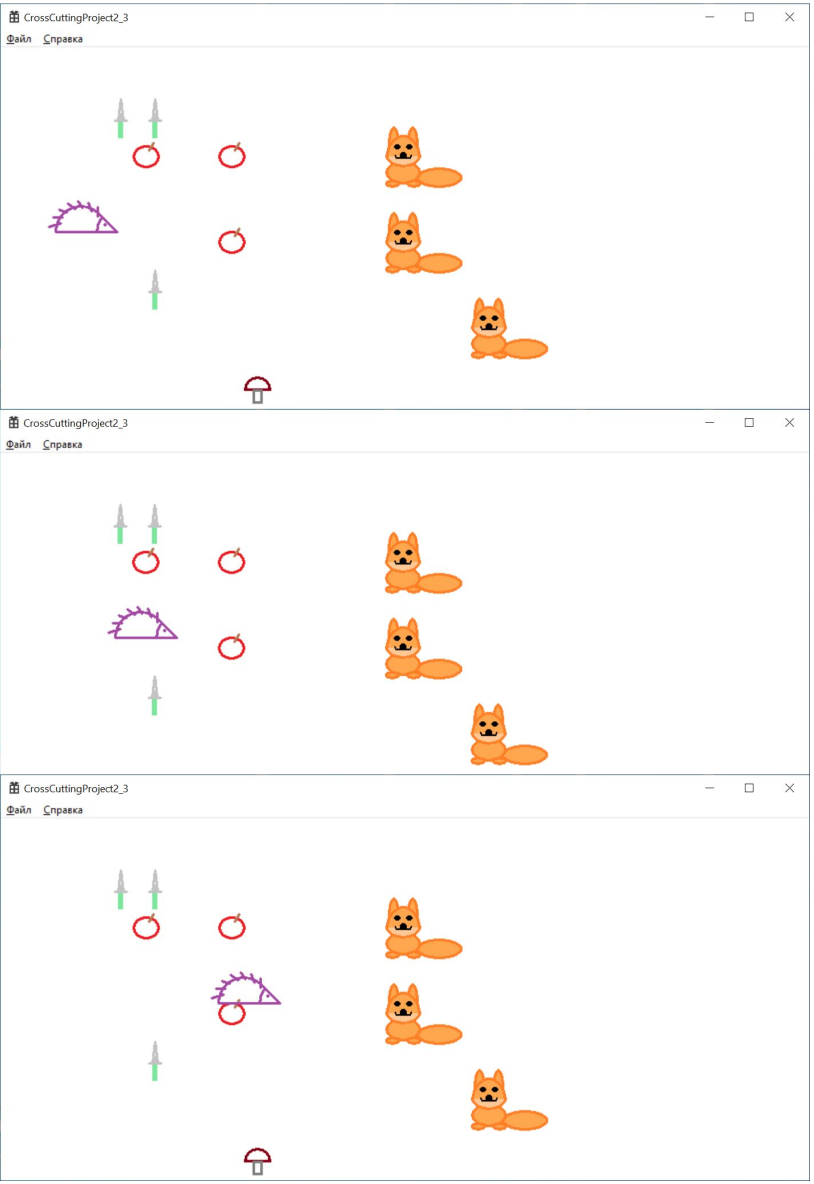


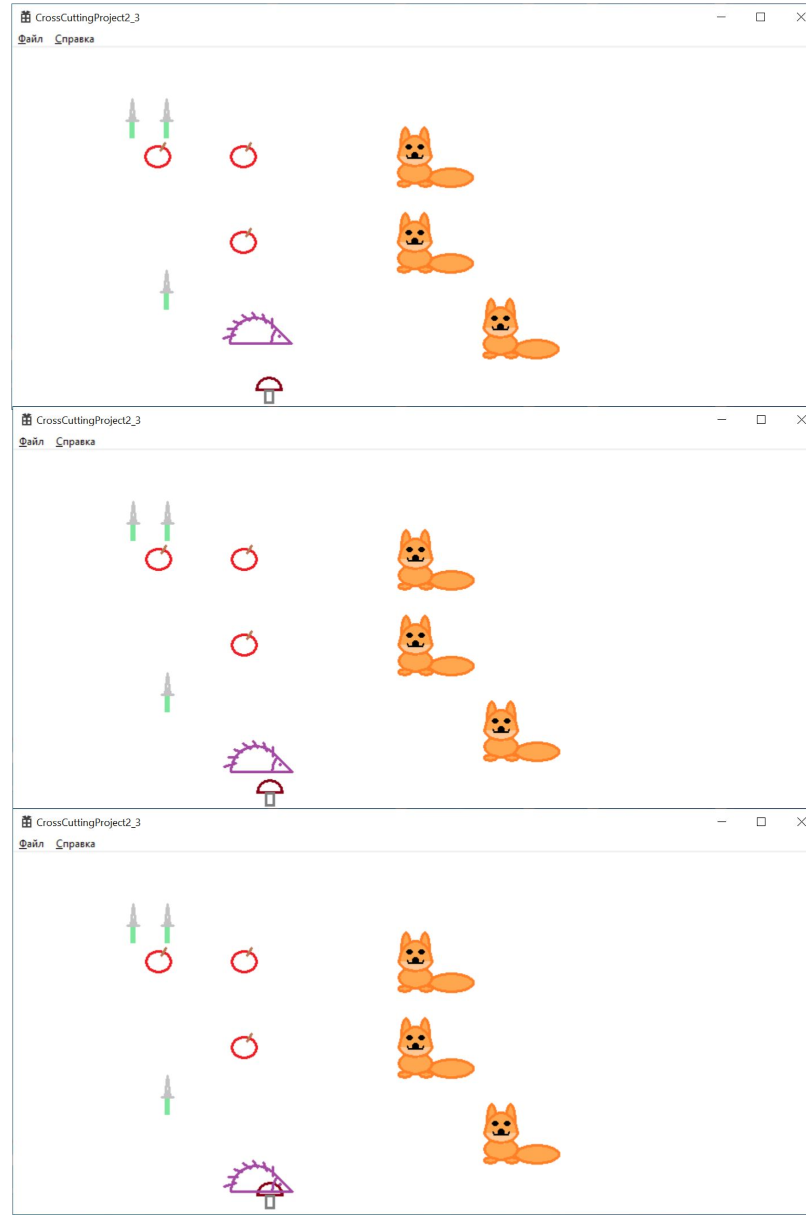
Задача 3. Реализация (3). Изменения в WndProc




```
358     break;
359
360     case WM_KEYDOWN:
361         switch (wParam)
362         {
363         case VK_UP:
364             HedgehogY -= 10;
365             if (MushroomIsEaten())
366                 PostQuitMessage(0);
367             InvalidateRect(hWnd, NULL, TRUE);
368             break;
369         case VK_DOWN:
370             HedgehogY += 10;
371             if (MushroomIsEaten())
372                 PostQuitMessage(0);
373             InvalidateRect(hWnd, NULL, TRUE);
374             break;
375
376         case VK_LEFT:
377             HedgehogX -= 10;
378             if (MushroomIsEaten())
379                 PostQuitMessage(0);
380             InvalidateRect(hWnd, NULL, TRUE);
381             break;
382         case VK_RIGHT:
383             HedgehogX += 10;
384             if (MushroomIsEaten())
385                 PostQuitMessage(0);
386             InvalidateRect(hWnd, NULL, TRUE);
387             break;
388         }
389         break;
390
391     case WM_PAINT:
392     {
393         PAINTSTRUCT ps;
394         HDC hdc = BeginPaint(hWnd, &ps);
395         // TODO: Добавьте сюда любой код прорисовки, использующий HDC...
396
397         DrawMushroom(hdc, MushroomX, MushroomY);
398
399         DrawKnife(hdc, 140, 85);
400         DrawKnife(hdc, 180, 85);
401         DrawKnife(hdc, 180, 285);
402
403         DrawApple(hdc, 170, 128);
404         DrawApple(hdc, 270, 128);
```

Задача 3. Работающее приложение





Задача 4. Презентация преподавателю получившегося приложения на занятии

Вам необходимо показывать ваши наработки по игре преподавателю после выполнения КАЖДОГО ЭТАПА проекта!

Если вы выполнили все этапы, но показали всё только в конце, то проект вам не засчитывается. ДАЖЕ ЕСЛИ ВЫ ДОКАЗАЛИ полное свое авторство – увы ☹️

ИТОГО по этапу 2

1. Все объекты игры перебрались в собственные функции. И теперь вы можете рисовать их в любом месте и в любом количестве!
2. Вы оставили на карте только одного героя и теперь управляете его перемещениями при помощи клавиатуры.
3. Вы оставили только один предмет, нужный герою. И проверяете достиг ли герой этого предмета или нет. И если он его достиг – то игра завершается!

У вас есть всё, чтобы идти дальше!

Сквозной проект – этап 3 (СПз)

Объект перемещается случайным образом.
Герою наносится ущерб.
Герой должен собрать много предметов.
Герой может подлечиваться.
Презентация итогов этапа 3 преподавателю.

Задачи ЭТАПА 3

Задача 1. Сделать автоматическое перемещение врага или предмета.
Задача 2. Герою наносится ущерб. В случае проигрыша игра заканчивается.
Задача 3. Герой собирает не один предмет, а множество.
Задача 4. Герой может подлечиться множеством предметов.
Задача 5. Презентация итогов этапа 3 преподавателю

Задача 1. Сделать автоматическое перемещение врага или предмета

Сделаем автоматическое перемещение лисы

Для этого нужно:

1) Завести глобальные переменные, в которых хранятся координаты лисы
2) В WM_PAINT при отрисовке лисы подставлять значения этих глобальных переменных
3) Добавить создание таймера при создании окна
4) При каждом срабатывании таймера вычислять перемещение лисы. И изменять координаты лисы.

Задача 1. Реализация (1). Глобальные переменные

CrossCuttingProject3_1.cpp*

CrossCuttingProject3_1 (Глобальная область)

```
1   // CrossCuttingProject3_1.cpp : Определяет точку входа для приложения.
2   //
3
4   #include "framework.h"
5   #include "CrossCuttingProject3_1.h"
6
7
8
9   #define MAX_LOADSTRING 100
10
11  // Глобальные переменные:
12  HINSTANCE hInst;                                // текущий экземпляр
13  WCHAR szTitle[MAX_LOADSTRING];                  // Текст строки заголовка
14  WCHAR szWindowClass[MAX_LOADSTRING];            // имя класса главного окна
15
16
17
18  // Координаты ёжика
19  int HedgehogX = 100;
20  int HedgehogY = 200;
21
22
23  // Координаты гриба
24  int MushroomX = 300;
25  int MushroomY = 400;
26
27
28  // Координаты лисы
29  int FoxX = 400;
30  int FoxY = 200;
```

Задача 1. Реализация (2). WM_PAINT

CrossCuttingProject3_1.cpp

CrossCuttingProject3_1 (Глобальная область)

```
469     case WM_PAINT:
470     {
471         PAINTSTRUCT ps;
472         HDC hdc = BeginPaint(hWnd, &ps);
473         // TODO: Добавьте сюда любой код прорисовки, использующий HDC...
474
475         DrawMushroom(hdc, MushroomX, MushroomY);
476
477         DrawKnife(hdc, 140, 85);
478         DrawKnife(hdc, 180, 85);
479         DrawKnife(hdc, 180, 285);
480
481         DrawApple(hdc, 170, 128);
482         DrawApple(hdc, 270, 128);
483         DrawApple(hdc, 270, 228);
484
485         DrawHedgehog(hdc, HedgehogX, HedgehogY);
486
487         DrawFox(hdc, FoxX, FoxY);
488
489         EndPaint(hWnd, &ps);
490     }
491     break;
492
493     case WM_DESTROY:
494         PostQuitMessage(0);
```

Задача 1. Реализация (3). Создание таймера

CrossCuttingProject3_1.cpp

CrossCuttingProject3_1 (Глобальная область)

```
1  // CrossCuttingProject3_1.cpp : Определяет точку входа для приложения.
2  //
3
4  #include "framework.h"
5  #include "CrossCuttingProject3_1.h"
6  #include <time.h>
7
8
9
10 #define MAX_LOADSTRING 100
11
12 // Глобальные переменные:
13 HINSTANCE hInst;                                // текущий экземпляр
14 WCHAR szTitle[MAX_LOADSTRING];                  // Текст строки заголовка
15 WCHAR szWindowClass[MAX_LOADSTRING];            // имя класса главного окна
16
17
```

CrossCuttingProject3_1.cpp

CrossCuttingProject3_1 (Глобальная область)

```
382 //
383 //
384 LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
385 {
386     switch (message)
387     {
388     case WM_COMMAND:
389     {
390         int wmId = LOWORD(wParam);
391         // Разобрать выбор в меню:
392         switch (wmId)
393         {
394         case IDM_ABOUT:
395             DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
396             break;
397         case IDM_EXIT:
398             DestroyWindow(hWnd);
399             break;
400         default:
401             return DefWindowProc(hWnd, message, wParam, lParam);
402         }
403     }
404     break;
405
406     // Сообщение о создании окна
407     case WM_CREATE:
408         // Перезапуск генератора случайных чисел
409         // функция time() объявлена в time.h
410         srand(time(NULL));
411
412         // Запуск таймера
413         // 1 - номер таймера
414         // 1000 - интервал срабатывания таймера 1000 мс = 1 сек
415         SetTimer(hWnd, 1, 1000, NULL);
416
417         break;
418
```

Задача 1. Реализация (4). Срабатывание таймера и перемещение лисы

CrossCuttingProject3_1.cpp

CrossCuttingProject3_1 (Глобальная область)

```
336     }
337
338     // Случайное перемещение лисы
339     void RandomMoveFox() {
340
341         // dx - изменение координаты x
342         // dy - изменение координаты y
343         // "rand() % 41" даст числа в диапазоне [0 .. 40] - всего 41 разное число
344         // "rand() % 41 - 20" даст числа в диапазоне [-20 .. 20] - всего 41 разное число
345         int dx = rand() % 41 - 20;
346         int dy = rand() % 41 - 20;
347
348         // Вычисляем новые координаты лисы
349         FoxX += dx;
350         FoxY += dy;
351     }
352
353
354 //
355 //  ФУНКЦИЯ: WndProc(HWND, UINT, WPARAM, LPARAM)
356 //
357 //  ЦЕЛЬ: Обрабатывает сообщения в главном окне.
358 //
359 //  WM_COMMAND  - обработать меню приложения
360 //  WM_PAINT    - Отрисовка главного окна
361 //  WM_DESTROY  - отправить сообщение о выходе и вернуться
362 //
363 //
364 LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
365 {
366     switch (message)
367     {
368     case WM_COMMAND:
369     {
370         int wmId = LOWORD(wParam);
371         // Разобрать выбор в меню:
372         switch (wmId)
373         {
374         case IDM_ABOUT:
375             DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
376             break;
```

CrossCuttingProject3_1.cpp*

CrossCuttingProject3_1 (Глобальная об

```
384         break;
385
386         // Сообщение о создании окна
387         case WM_CREATE:
388             // Перезапуск генератора случайных чисел
389             // функция time() объявлена в time.h
390             srand(time(NULL));
391
392             // Запуск таймера
393             // 1 - номер таймера
394             // 1000 - интервал срабатывания таймера 1000 мс = 1 сек
395             SetTimer(hWnd, 1, 1000, NULL);
396
397             break;
398
399         // Сообщение "Таймер сработал"
400         case WM_TIMER:
401
402             // Перемещаем лису
403             RandomMoveFox();
404
405             // Заставляем все перерисовать
406             InvalidateRect(hWnd, NULL, TRUE);
407
408             break;
409
410         case WM_KEYDOWN:
411             switch (wParam)
412             {
413             case VK_UP:
414                 HedgehogY -= 10;
415                 if (MushroomIsEaten())
416                     PostQuitMessage(0);
417                 InvalidateRect(hWnd, NULL, TRUE);
418                 break;
```

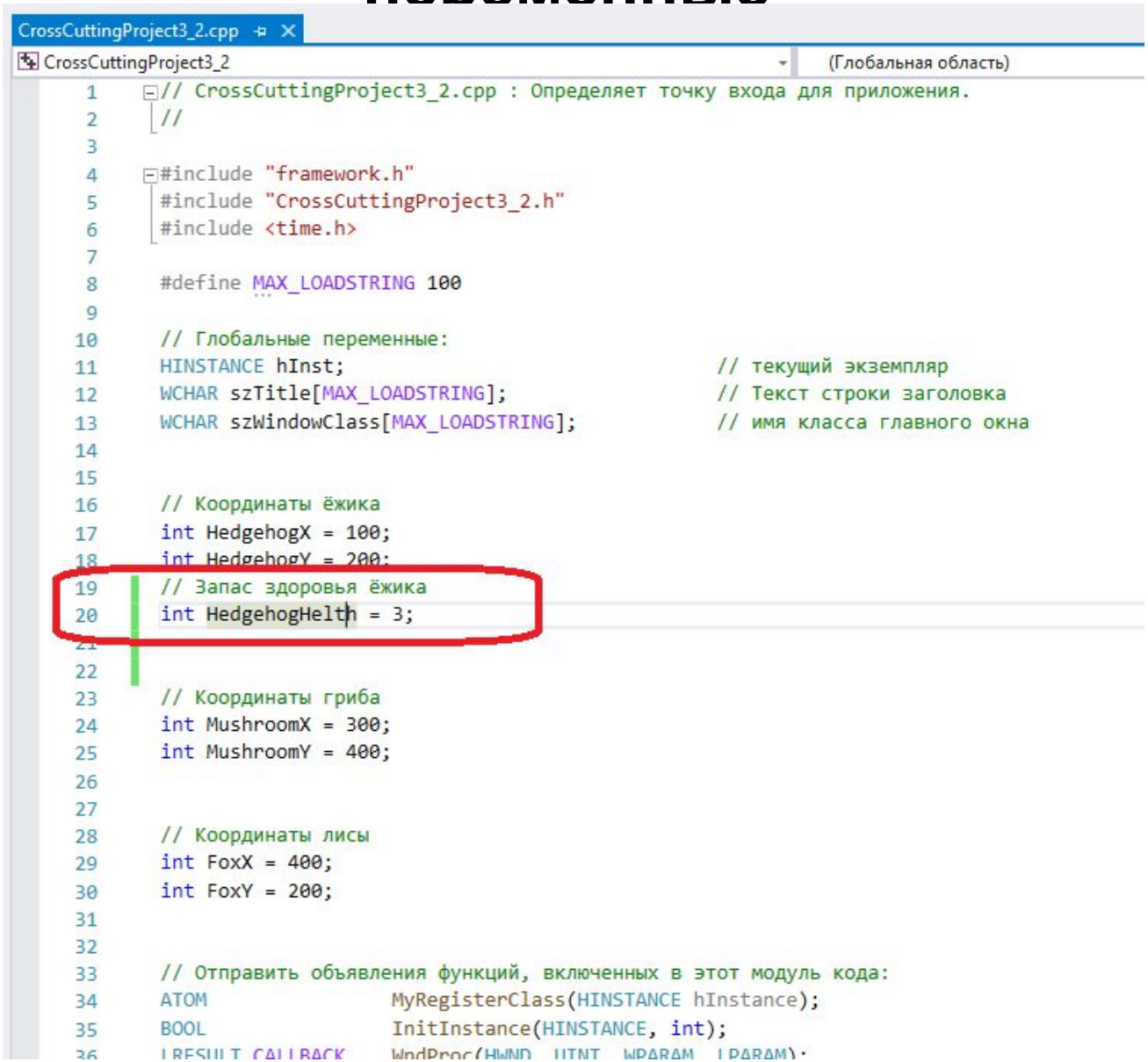
Задача 2. Герою наносится ущерб. В случае проигрыша игра заканчивается.

Каждую секунду (при срабатывании таймера – сразу после перемещения) проверяем может ли лиса съесть ёжика. Если может – то уменьшаем его запас здоровья. Если здоровья не осталось – игра заканчивается.

Для реализации этой задачи нужно:

1) Завести глобальную переменную, в которой хранится количество здоровья ёжика
2) При срабатывании таймера (событие WM_TIMER) пытаемся съесть ёжика
3) После попытки съесть ёжика проверяем – есть у него еще здоровье или нет. Если здоровья нет – то завершаем игру

Задача 2. Реализация (1). Глобальные переменные



Задача 2. Реализация (2). Пытаемся съесть ёжика

```
350
351  void TryToEatHedgehog() {
352
353      if (HedgehogX - 20 < FoxX
354          && HedgehogX + 20 > FoxX
355          && HedgehogY - 20 < FoxY
356          && HedgehogY + 20 > FoxY) {
357
358          HedgehogHelth--;
359
360      }
361
362  }
363
364
365
366
367  //
368  //  ФУНКЦИЯ: WndProc(HWND, UINT, WPARAM, LPARAM)
369  //
370  //  ЦЕЛЬ: Обрабатывает сообщения в главном окне.
371  //
372  //  WM_COMMAND  - обработать меню приложения
373  //  WM_PAINT    - Отрисовка главного окна
374  //  WM_DESTROY  - отправить сообщение о выходе и вернуться
375  //
376  //
377  LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lPara
378  {
```

CrossCuttingProject3_2.cpp

CrossCuttingProject3_2

```
409
410          break;
411
412          // Сообщение "Таймер сработал"
413      case WM_TIMER:
414
415          // Перемещаем лису
416          RandomMoveFox();
417
418          TryToEatHedgehog();
419
420          if (HedgehogHelth == 0) {
421              PostQuitMessage(0);
422          }
423
424          // Заставляем все перерисовать
425          InvalidateRect(hWnd, NULL, TRUE);
426          break;
427
428
429
430
431      case WM_KEYDOWN:
432          switch (wParam)
433          {
434          case VK_UP:
435              HedgehogY -= 10;
436              if (MushroomIsEaten())
437                  PostQuitMessage(0);
438              InvalidateRect(hWnd, NULL, TRUE);
439              break;
```

Задача 2. Реализация (3). Если здоровья больше нет – завершаем игру

```
411
412         // Сообщение "Таймер сработал"
413     case WM_TIMER:
414
415         // Перемещаем лису
416         RandomMoveFox();
417
418         TryToEatHedgehog();
419
420         if (HedgehogHelth == 0) {
421             PostQuitMessage(0);
422         }
423
424         // Заставляем все перерисовать
425         InvalidateRect(hWnd, NULL, TRUE);
426         break;
427
428
```

Задача 3. Герой собирает не один предмет, а множество

Ежик должен собрать не 1 гриб, а 12 чтобы успешно закончить игру.
1, 2 или 3 гриба можно было бы хранить в наборе отдельных переменных, но если их 12 – код обработки будет очень большим и сложным. Поэтому использовать массивы, чтобы это реализовать.

Для этого нужно:

1) Глобальные переменные, отвечающие за координаты гриба переделываем в массивы. Поскольку грибы уже собранные не должны отрисовываться, то нужно для каждого гриба еще добавить переменную отражающую статус «Этот гриб не собран!»
2) В WM_PAINT при отрисовке гриба будем использовать массивы.
3) При проверке совпадения координат ежика с грибами также будем использовать массивы.
4) При проверке выполнил ли ежик свою работу и можно ли уже завершать игру также будем использовать массивы.

Задача 3. Реализация (1). Глобальные переменные

CrossCuttingProject3_3.cpp*

CrossCuttingProject3_3 | (Глобальная область)

```
1   // CrossCuttingProject3_3.cpp : Определяет точку входа для приложения.
2   //
3
4   #include "framework.h"
5   #include "CrossCuttingProject3_3.h"
6   #include <time.h>
7
8
9
10  #define MAX_LOADSTRING 100
11
12  // Глобальные переменные:
13  HINSTANCE hInst;                                // текущий экземпляр
14  WCHAR szTitle[MAX_LOADSTRING];                  // Текст строки заголовка
15  WCHAR szWindowClass[MAX_LOADSTRING];            // имя класса главного окна
16
17
18  // Координаты ёжика
19  int HedgehogX = 100;
20  int HedgehogY = 200;
21  // Запас здоровья ёжика
22  int HedgehogHelth = 3;
23
24
25  // Координаты грибов
26  // количество грибов
27  #define NUM_MUSHROOMS 12
28  // координата X каждого гриба
29  int MushroomX[NUM_MUSHROOMS] = { 200, 220, 240, 260,      300, 400, 420, 440,      460, 480, 500, 520 };
30  // координата Y каждого гриба
31  int MushroomY[NUM_MUSHROOMS] = { 400, 350, 300, 250,      300, 200, 100, 200,      100, 140, 120, 100 };
32  // видим ли гриб? (для каждого гриба!)
33  int MushroomVisible[NUM_MUSHROOMS] = { 1, 1, 1, 1,        1, 1, 1, 1,              1, 1, 1, 1 };
34
35
36  // Координаты лисы
37  int FoxX = 400;
38  int FoxY = 200;
39
```

Задача 3. Реализация (2). Отрисовка

```
361
362  // Отрисовка всех видимых грибов
363  void DrawMushrooms(HDC hdc) {
364
365      int i = 0;
366      do {
367
368          // если i гриб виден (не съеден)
369          if (MushroomVisible[i]) {
370              // отрисовываем i гриб
371              DrawMushroom(hdc, MushroomX[i], MushroomY[i]);
372          }
373
374          ++i;
375      } while (i < NUM_MUSHROOMS);
376
377  }
378
```

```
519
520      case WM_PAINT:
521      {
522          PAINTSTRUCT ps;
523          HDC hdc = BeginPaint(hWnd, &ps);
524          // TODO: Добавьте сюда любой код прорисовки, использующий HDC...
525
526          DrawMushrooms(hdc);
527
528          DrawKnife(hdc, 140, 85);
529          DrawKnife(hdc, 180, 85);
530          DrawKnife(hdc, 180, 285);
531
532          DrawApple(hdc, 170, 128);
533          DrawApple(hdc, 270, 128);
534          DrawApple(hdc, 270, 228);
535
536          DrawHedgehog(hdc, HedgehogX, HedgehogY);
537
538          DrawFox(hdc, FoxX, FoxY);
539
540          EndPaint(hWnd, &ps);
541      }
542      break;
543
```

Задача 3. Реализация (3). Поедание грибов

```
// пытаемся съесть гриб
void TryToEatMushroom() {

    // пробегаем по всем грибам
    int i = 0;
    do {

        // если i гриб виден
        if (MushroomVisible[i]) {

            // если ежик может съесть i гриб
            if (HedgehogX - 20 < MushroomX[i]
                && HedgehogX + 20 > MushroomX[i]
                && HedgehogY - 20 < MushroomY[i]
                && HedgehogY + 20 > MushroomY[i]) {

                // i гриб делаем невидимым - его съели!!!
                MushroomVisible[i] = 0;
            }
        }
        // переходим к следующему грибу
        ++i;

    } while (i < NUM_MUSHROOMS); // цикл завершаем после обхода всех NUM_MUSHROOMS грибов!!!

}
```

```
    // Сообщение "Клавишу нажали"
    case WM_KEYDOWN:
        switch (wParam)
        {
        case VK_UP:
            HedgehogY -= 10;
            InvalidateRect(hWnd, NULL, TRUE);
            break;
        case VK_DOWN:
            HedgehogY += 10;
            InvalidateRect(hWnd, NULL, TRUE);
            break;
        case VK_LEFT:
            HedgehogX -= 10;
            InvalidateRect(hWnd, NULL, TRUE);
            break;
        case VK_RIGHT:
            HedgehogX += 10;
            InvalidateRect(hWnd, NULL, TRUE);
            break;
        }

        TryToEatMushroom();
        if (CountVisibleMushrooms() == 0) {
            PostQuitMessage(0);
        }
        break;
```

Задача 3. Реализация (4). Окончание игры

```
406
407
408   // подсчитать количество видимых (не съеденных) грибов
409   int CountVisibleMushrooms() {
410       // счетчит видимых грибов
411       int count = 0;
412       int i = 0;
413       do {
414
415           // если i гриб видим - счетчик увеличиваем на 1
416           if (MushroomVisible[i]) {
417               ++count;
418           }
419
420           ++i;
421       } while (i < NUM_MUSHROOMS);
422
423       // возвращаем количество видимых (не съеденных) грибов
424       return count;
425   }
426
427
```

```
489
490
491       // Сообщение "Клавишу нажали"
492       case WM_KEYDOWN:
493           switch (wParam)
494           {
495           case VK_UP:
496               HedgehogY -= 10;
497               InvalidateRect(hWnd, NULL, TRUE);
498               break;
499           case VK_DOWN:
500               HedgehogY += 10;
501               InvalidateRect(hWnd, NULL, TRUE);
502               break;
503           case VK_LEFT:
504               HedgehogX -= 10;
505               InvalidateRect(hWnd, NULL, TRUE);
506               break;
507           case VK_RIGHT:
508               HedgehogX += 10;
509               InvalidateRect(hWnd, NULL, TRUE);
510               break;
511           }
512
513
514           TryToEatMushroom();
515           if (CountVisibleMushrooms() == 0) {
516               PostQuitMessage(0);
517           }
518           break;
519
520
```

Задача 4. Герой может подлечиться множеством предметов

В игре «Жизнь ёжика» есть предметы «Яблоки», которые для ёжика имеют оздоровительное свойство. Если ёжик съедает яблоко, то его здоровье увеличивается на 1. Реализуем возможность оздоровления ёжика.

Для этого нужно:

1) Глобальные переменные, отвечающие за координаты яблок и за их видимость. Поскольку у нас будет множество яблок, то будем использовать массив.
2) В WM_PAINT при отрисовке яблок будем использовать массивы.
3) Реализуем проверку совпадения координат ежика с яблоками. Если координаты совпали – то будем увеличивать запас здоровья ежика, а яблоко делаем невидимым.

Задача 4. Реализация (1). Глобальные переменные

CrossCuttingProject3_4.cpp*

CrossCuttingProject3_4 (Глобальная область)

```
1   // CrossCuttingProject3_4.cpp : Определяет точку входа для приложения.
2   //
3
4   #include "framework.h"
5   #include "CrossCuttingProject3_4.h"
6   #include <time.h>
7
8   #define MAX_LOADSTRING 100
9
10  // Глобальные переменные:
11  HINSTANCE hInst;                                // текущий экземпляр
12  WCHAR szTitle[MAX_LOADSTRING];                  // Текст строки заголовка
13  WCHAR szWindowClass[MAX_LOADSTRING];            // имя класса главного окна
14
15
16  // Координаты ёжика
17  int HedgehogX = 100;
18  int HedgehogY = 200;
19  // Запас здоровья ёжика
20  int HedgehogHelth = 3;
21
22
23  // Координаты грибов
24  // количество грибов
25  #define NUM_MUSHROOMS 12
26  // координата X каждого гриба
27  int MushroomX[NUM_MUSHROOMS] = { 200, 220, 240, 260,      300, 400, 420, 440,       460, 480, 500, 520 };
28  // координата Y каждого гриба
29  int MushroomY[NUM_MUSHROOMS] = { 400, 350, 300, 250,      300, 200, 100, 200,       100, 140, 120, 100 };
30  // виден ли гриб? (для каждого гриба!)
31  int MushroomVisible[NUM_MUSHROOMS] = { 1, 1, 1, 1,        1, 1, 1, 1,               1, 1, 1, 1 };
32
33
34  // Координаты яблок
35  // количество яблок
36  #define NUM_APPLES 3
37  // координата X каждого яблока
38  int AppleX[NUM_APPLES] = { 170, 270, 270 };
39  // координата Y каждого яблока
40  int AppleY[NUM_APPLES] = { 128, 128, 228 };
41  // видно ли яблоко? (для каждого яблока!)
42  int AppleVisible[NUM_APPLES] = { 1, 1, 1};
43
44
45  // Координаты лисы
46  int FoxX = 400;
47  int FoxY = 200;
```

Задача 4. Реализация (2). Отрисовка

```
435
436
437  // Отрисовка всех видимых яблок
438  void DrawApples(HDC hdc) {
439
440      int i = 0;
441      do {
442
443          // если i яблоко видно (не съедено)
444          if (AppleVisible[i]) {
445              // отрисовываем i яблоко
446              DrawApple(hdc, AppleX[i], AppleY[i]);
447          }
448
449          ++i;
450      } while (i < NUM_APPLES);
451
452  }
453
454
```

```
547
548
549      case WM_PAINT:
550      {
551          PAINTSTRUCT ps;
552          HDC hdc = BeginPaint(hWnd, &ps);
553          // TODO: Добавьте сюда любой код прорисовки, использующий HDC...
554
555          DrawMushrooms(hdc);
556
557          DrawKnife(hdc, 140, 85);
558          DrawKnife(hdc, 180, 85);
559          DrawKnife(hdc, 180, 285);
560
561          //DrawApple(hdc, 170, 128);
562          //DrawApple(hdc, 270, 128);
563          //DrawApple(hdc, 270, 228);
564          DrawApples(hdc);
565
566          DrawHedgehog(hdc, HedgehogX, HedgehogY);
567
568          DrawFox(hdc, FoxX, FoxY);
569
570          EndPaint(hWnd, &ps);
571      }
572      break;
573
574
```

Задача 4. Реализация (3). Поедание яблок

```
454
455
456  // пытаемся съесть яблоко
457  void TryToEatApple() {
458
459      // пробегаем по всем яблокам
460      int i = 0;
461      do {
462
463          // если i яблоко видно
464          if (AppleVisible[i]) {
465
466              // если ежик может съесть i яблоко
467              if (HedgehogX - 20 < AppleX[i]
468                  && HedgehogX + 20 > AppleX[i]
469                  && HedgehogY - 20 < AppleY[i]
470                  && HedgehogY + 20 > AppleY[i]) {
471
472                  // i яблоко делаем невидимым - его съели!!!
473                  AppleVisible[i] = 0;
474                  // поправляем здоровье ежика
475                  ++HedgehogHelth;
476
477              }
478          }
479          // переходим к следующему яблоку
480          ++i;
481
482      } while (i < NUM_APPLES); // цикл завершаем после обхода всех NUM_APPLES яблок!!!
483
484  }
485
486
```

```
548
549
550          // Сообщение "Клавишу нажали"
551      case WM_KEYDOWN:
552          switch (wParam)
553          {
554          case VK_UP:
555              HedgehogY -= 10;
556              InvalidateRect(hWnd, NULL, TRUE);
557              break;
558          case VK_DOWN:
559              HedgehogY += 10;
560              InvalidateRect(hWnd, NULL, TRUE);
561              break;
562          case VK_LEFT:
563              HedgehogX -= 10;
564              InvalidateRect(hWnd, NULL, TRUE);
565              break;
566          case VK_RIGHT:
567              HedgehogX += 10;
568              InvalidateRect(hWnd, NULL, TRUE);
569              break;
570          }
571
572
573          TryToEatMushroom();
574          if (CountVisibleMushrooms() == 0) {
575              PostQuitMessage(0);
576          }
577
578          TryToEatApple();
579          break;
580
581
```

Задача 5. Презентация преподавателю получившегося приложения на занятии

Вам необходимо показывать ваши наработки по игре преподавателю после выполнения КАЖДОГО ЭТАПА проекта!

Если вы выполнили все этапы, но показали всё только в конце, то проект вам не засчитывается. ДАЖЕ ЕСЛИ ВЫ ДОКАЗАЛИ полное свое авторство – увы ☹️

ИТОГО по этапу 3

1. Ваши предметы научились самостоятельно перемещаться.
2. Вашему герою наносится ущерб.
3. Но при этом ваш герой научился поправлять свое здоровье.
4. Предметов стало много, поскольку вы стали использовать массивы для их хранения.

У вас уже есть игра, в которую можно поиграть.
Идем дальше!

ИТОГО по сквозному проекту

1. Обсудили что такое ОС, интерфейс, GUI, API, Windows API (WinAPI)
2. Узнали как создать WinAPI приложение в VS.
3. Узнали/повторили декартову систему координат.
4. Узнали про экранную систему координат.
5. Узнали как нарисовать рисунок из линий.
6. Узнали как задать стиль, цвет и толщину у пера.
7. Узнали как нарисовать прямоугольник и эллипс.
8. Узнали как создать кисть для рисования фигур.
9. Узнали что нужно делать в ЛР4 и ЛР5.

Источники информации

- Рисование геометрических фигур - https://www.frolov-lib.ru/books/bsp/v14/ch2_3.htm

- КАК рисовать в Win32 API? -http://radiofront.narod.ru/htm/prog/htm/winda/api/paint.html