Документація до лабораторної роботи №2  
з дисципліни Основи Об’єктно орієнтоване програмування

Студента факультету   
комп'ютерних наук та кібернетики  
Групи ІПС-22  
Дехтяренка Михайла

**Мова реалізації С++**

В цій лабораторній роботі я розгляну елементи керування в проекті Game TD.   
 В проекті представлено:  
 3 кнопки які змінюють стан гри. Починають її, зупиняють та відкривають меню текстур.  
 Перенос веж з панелі на поле та між полями.   
 Можливість вибрати який зовнішній вигляд вежі або поля хоче користувач, що додає ще 6(12 з урахуванням того, що зовнішній вигляд клікабельний).   
 Можливість вимкнути гру нажавши на esс.   
  
Звідси випливає що в програмі є:  
- Більше ніж 15(з урахуванням подвійних кнопок текстур) елементів керування  
- Більше ніж 10 обробників подій   
-3 екрани (start\stop, skins, game over)

Модулі програми

**Кожну кнопку в програмі можна описати таким алгоритмом:**

|  |
| --- |
| glDisable(GL\_COLOR);  glEnable(GL\_TEXTURE\_2D);  glBindTexture(GL\_TEXTURE\_2D, n); // функція, яка загружає текстуру з id n  glBegin(GL\_QUADS);  glTexCoord2f(textures[0][0], textures[0][1]); // Створює першу точку текстури  glVertex2d(x , y); // задаємо першу координату  glTexCoord2f(textures[1][0], textures[1][1]); // другу точку текстури  glVertex2d(x + width, y); // задаємо другу координату  glTexCoord2f(textures[2][0], textures[2][1]); // третью точку текстури  glVertex2d(x + width, y + height); // задаємо третью координату  glTexCoord2f(textures[3][0], textures[3][1]); // четверту точку текстури  glVertex2d(x, y + height); // задаємо четверту координату  glEnd();  glDisable(GL\_TEXTURE\_2D);  glEnable(GL\_COLOR); |

Обробником події тут виступає елемент функції Mouse:

|  |
| --- |
| if (xx > x && xx < x + width && yy > y && yy < y + heigh )  // якщо координати миші знаходяться у діапазоні кнопки і в грі встановлен вірний статус, то натискання на кнопку щось змінить.   {  event;  glutPostRedisplay();  } |

**Розташування веж:**

Воно будується з використанням двох функцій: Mouse та MouseMotion   
Ця функція перевіряє, чи була взята якась вежа, якщо так, то їде перевірка на те:  
- що місце розташування вежі не є дорогою, статус гри є активний чи не розташована вежа далі 26 елемента матриці (щоб не було багів з відображенням)

- цикл проходить кожну вежу, якщо координати вибраної вежі співпадають з координатами іншої та це не є поточна вежа.   
  
Якщо одна з цих умов не виповнена, вежа повертається на початкову точку та деактивується

|  |
| --- |
| if (state == GLUT\_UP)  {  if (currentTower != -1)  {  if (pRoad->a[towers[currentTower].n][towers[currentTower].m] == 1 || status > 0 || towers[currentTower].m > 26)  {  towers[currentTower].m = currentTower;  towers[currentTower].n = 1;  towers[currentTower].active = true;  }  for (int i = 0; i < nTowers; i++)  {  if ((towers[currentTower].m == towers[i].m && towers[currentTower].n == towers[i].n && currentTower != i))  {  towers[currentTower].m = currentTower;  towers[currentTower].n = 1;  towers[currentTower].active = true;  break;  }  };  }    ButtonStatus = -1;  currentTower = -1;  }  break;  default:  break;  } |

Функція завдяки якій візуально створено перенесення веж

|  |
| --- |
| void MouseMotion(int x, int y)  {  if (ButtonStatus == 1)  {  int xx, yy;  xx = x;  yy = h - y;  int im, jm;  im = xx / 40;  jm = yy / 40;  if (currentTower != -1)  {  towers[currentTower].m = im;  towers[currentTower].n = jm;  towers[currentTower].active = false;  }  glutPostRedisplay();  }  } |

**Вимкнення програми**

Проста функція, яка вимикає програму, якщо нажати на esc

|  |
| --- |
| void keyb(unsigned char key, int x, int y)  {  const unsigned char esc = 27;  if (key == esc)  {  exit(0);  }  } |