|  |
| --- |
| 윈도우즈 프로그래밍 |

-과제-

12주차

윈도우 폼 : 스네이크 게임

20144638

이한범

# 예제 1번

수치적 업다운 컨트롤(numericUpDown), 프로그레스 바(ProgressBar), 타이머 컴포넌트로 화면이 구성된 스네이크 게임을 작성하시오.

-규칙으로는 수치적 업다운 컨트롤로 레벨을 설정, OK 버튼을 누르게 되면 시작하게된다.

-방향키로 스네이크를 조정한다.

-스네이크의 머리가 테두리나 스네이크의 몸과 충돌시 게임종료.

-빨간색 사각형은 item으로, 먹으면 길이가 길어지고 랜덤한 위치에 나타난다.

-아이템을 먹으면 프로그레스바가 1증가하고, 10 이상으로 증가시 레벨 증가.

-레벨이 증가할수록 속도가 빨라진다.

1. 윈도우 폼 구성도

|  |
| --- |
|  |

1. 작성한 코드내용

# 항목 더블클릭 이벤트

|  |
| --- |
| namespace Snake  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  }  static Rectangle[] snake = new Rectangle[100];  static int snake\_Count = 5;  static char direction = ' '; //뱀의 방향  static int item\_X=-1, item\_Y, item\_Count=0;  //아이템 위치 배치메서드  public void setItem()  {  bool conflict = true;  while (true)  {  //랜덤한 값 설정  Random r = new Random();  item\_X = r.Next(24) \* 10;  item\_Y = r.Next(24) \* 10;  for (int i = 0; i < snake\_Count; i++)  {  if (snake[i].X == item\_X && snake[i].Y == item\_Y)  conflict = false;  }  if (conflict == true) break;  }  }  //이미지 애니메이션을 위한 timer 메서드  private void timer1\_Tick(object sender, EventArgs e)  {  //만일, 뱀의 머리가 아이템에 닿았을 경우  if (snake[0].X == item\_X && snake[0].Y == item\_Y)  {  item\_Count++; // 먹은 아이템 수 증가  progressBar1.Value++; // 프로그레스 바에 한칸 추가  setItem(); //아이템 재배치  snake\_Count++; //스네이크 길이 증가  place.Invalidate(); // 패널을 다시 그리는 메서드  }  //레벨 업  if (item\_Count == progressBar1.Maximum)  {  numericUpDown1.Value++;  //레벨이 오른상태로 재시작하기 위해 버튼클릭  button1\_Click(sender, e);  }  //패널 범위를 벗어났을 때  if (snake[0].X > place.Width - 10 || snake[0].X < 0 || snake[0].Y > place.Height - 10 || snake[0].Y < 0)  {  timer1.Stop();  MessageBox.Show("Game Over..");  }  for (int i = 1; i < snake\_Count; i++)  {  if (snake[0].X == snake[i].X && snake[0].Y == snake[i].Y)  {  timer1.Stop();  MessageBox.Show("You die");  }  }  place.Invalidate();  }  private void controler\_KeyUp(object sender, KeyEventArgs e)  {  switch (e.KeyCode)  {  case Keys.Left:  if (direction == 'u' || direction == 'd')  direction = 'l';  break;  case Keys.Right:  if (direction == 'u' || direction == 'd')  direction = 'r';  break;  case Keys.Up:  if (direction == 'l' || direction == 'r')  direction = 'u';  break;  case Keys.Down:  if (direction == 'l' || direction == 'r')  direction = 'd';  break;  }  }  //Key 이벤트를 위한 포커스 대상  private void controler\_KeyUp\_1(object sender, KeyEventArgs e)  {  switch (e.KeyCode)  {  case Keys.Left:  if (direction == 'u' || direction == 'd')  direction = 'l';  break;  case Keys.Right:  if (direction == 'u' || direction == 'd')  direction = 'r';  break;  case Keys.Up:  if (direction == 'l' || direction == 'r')  direction = 'u';  break;  case Keys.Down:  if (direction == 'l' || direction == 'r')  direction = 'd';  break;  }  }  //패널 내에 그림을 그리는 메서드  private void place\_Paint(object sender, PaintEventArgs e)  {  //그래픽 객체를 이용하여 그리기 작업  Graphics g = e.Graphics;  int i;  if (direction == ' ');  //왼쪽방향  else if (direction == 'l')  {  for (i = snake\_Count - 1; i > 0; i--)  snake[i] = snake[i - 1];  snake[0].X = snake[1].X - 10;  }  //오른쪽방향  else if (direction == 'r')  {  for (i = snake\_Count - 1; i > 0; i--)  snake[i] = snake[i - 1];  snake[0].X = snake[1].X + 10;  }  //윗방향  else if (direction == 'u')  {  for (i = snake\_Count - 1; i > 0; i--)  snake[i] = snake[i - 1];  snake[0].Y = snake[1].Y - 10;  }  //아래방향  else if (direction == 'd')  {  for (i = snake\_Count - 1; i > 0; i--)  snake[i] = snake[i - 1];  snake[0].Y = snake[1].Y + 10;  }  //실제로 화면에 그려주는 부분  if (item\_X != -1)  {  Rectangle item = new Rectangle(item\_X, item\_Y, 10, 10);  g.FillRectangle(System.Drawing.Brushes.Red, item);  }  for (i = 0; i < snake\_Count; i++)  g.FillRectangle(System.Drawing.Brushes.Black, snake[i]);  controler.Focus();  }  //OK 버튼 클릭 이벤트  //클릭 될 시, 게임이 시작되며 아이템이 배치된다.  //각종 수치값과 이미지를 초기화. 단, 레벨은 초기화 시키지 않는다.  private void button1\_Click(object sender, EventArgs e)  {  snake[0] = new Rectangle(200, 200, 10, 10);  snake[1] = new Rectangle(210, 200, 10, 10);  snake[2] = new Rectangle(220, 200, 10, 10);  snake[3] = new Rectangle(230, 200, 10, 10);  snake[4] = new Rectangle(240, 200, 10, 10);  snake\_Count = 5; //새로시작시 뱀의 크기 초기화  timer1.Interval = 600 / (int)numericUpDown1.Value;  progressBar1.Value = 0;  item\_Count = 0;  direction = 'l';  setItem();  timer1.Start();  place.Invalidate();  controler.Focus();  }  }  } |

▲ 해당 폼은 6개의 필드를 가진다. snake는 스네이크를 그리기 위한 Rectangle 배열로, 최대 100의 길이까지 증가한다. snake\_Count는 스네이크의 길이를 저장하는 필드이다. direction은 스네이크의 이동방향을 값으로 가지는 필드이며, item\_X,Y,Count는 아이템의 위치 및 개수값을 가진다.

▲ setItem() : 아이템의 위치를 랜덤하게 설정하고, 충돌감지를 구현. 아이템의 X,Y 위치와 snake의 첫번째 네모의 X, Y와 동일하면 사라진다.

▲ timer1\_Tick() : 타이머 이벤트. 이미지 애니메이션을 위해 정의한 메서드. 뱀의 머리가 아이템에 닿을 경우에 아이템 먹은 수를 1 늘리고 프로그레스 바를 1칸 증가시킨다. 그 이후, setItem() 메서드를 다시 호출하여 아이템 위치를 재배치한다. 또한 스네이크의 길이도 증가시킨다.

마지막으로 place(패널)의 Invalidate() 메서드를 호출하여 화면을 다시 그린다.

아이템의 수가 프로그레스 바의 최대치와 동일해지면 레벨을 증가시키고 버튼 클릭 이벤트(button1\_Click)를 발생시켜 게임을 재시작하며, 패널범위를 벗어났을 때나 스네이크 몸통에 부딪혔을 때, Game Over 메시지 박스를 띄운다.

시간이 지날 때 마다 패널을 다시그려준다.

▲ controler\_KeyUp() : 키 이벤트 컨트롤을 위하여 포커스하는 요소로 사용한다. 방향키를 누름으로써 해당 방향으로 이동하도록 방향설정을 하는 메서드다이다.

▲ place\_Paint() : 패널 내에 그림을 그리는 메서드다이다. 그래픽 객체를 이용하여 그리기 작업을 하며, 방향키에 따라 스네이크들을 해당 위치로 이동시키는 작업을 한다.

OK 버튼을 누르기 전까지 화면에 그려지지 않으며, Controler 도구에 포커스를 맞추어 키 이벤트를 발생시킬 수 있도록 한다.

▲ button1\_Click() : OK 버튼을 누르면 발생하는 이벤트로, 스네이크 위치를 초기화하고, timer의 시간단위를 설정하며 프로그레스 바의 값, 아이템 갯수 값을 초기화한다.

setItem()을 호출하여 아이템 위치를 초기화 시키고 타이머를 시작시킨 후에 패널에 그림을 그린다. place\_Paint() 메서드와 마찬가지로 controler에 포커싱을 맞춰준다.

3) 설정

# Timer

|  |
| --- |
|  |

▲ 기본 타이머의 주기(interval)은 100(0.1초) 이며, 폼을 실행 후 OK버튼을 누를 시, 주기를 600 / 레벨수 로 변경해줍니다.

4) 실행결과

|  |
| --- |
|  |

▲ OK 버튼을 누를 시에 게임이 시작되며 아이템을 먹을 때 마다 프로그레스 바의 값이 올라감을 확인 할 수 있다.

▲ 만일 벽에 부딪히거나, 자기 몸에 부딪히게 될 경우 Game over 메시지 창을 띄우게 된다.