Gravity Panic

What if when you fell, you didn't just fall down? What if you could fall up, or sideways? What if you could control the direction you fell? This is exactly the disorienting scenario that you're in when you play Gravity Panic.

In Gravity Panic you use the arrow keys to move. Pressing up jumps, but up depends on the direction gravity is pulling you. Use the 'W', 'A', 'S', and 'D' keys to adjust gravity. You can change gravity mid-fall, but only once until you hit the ground again. Try to collect all the coins bug don't fall off the edges or you fall into the abyss.

If you think a level is impossible youc an press 'R' to create a new level and start over.

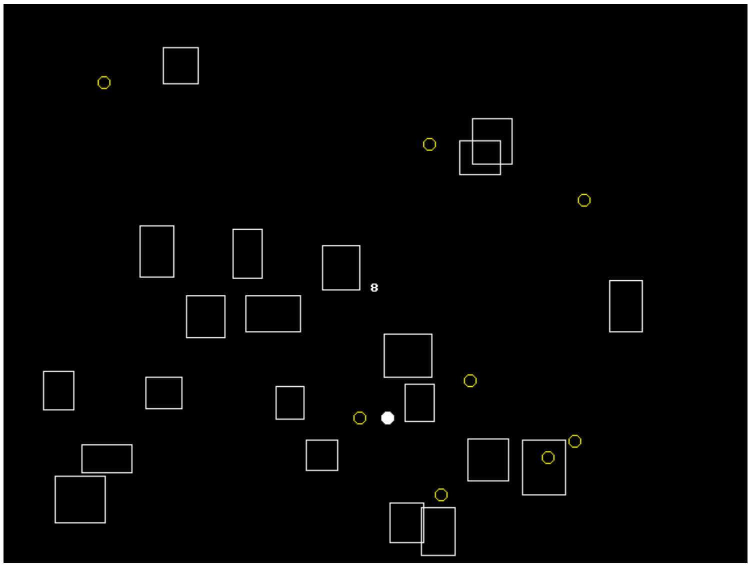
Gravity Panic is by is by Jacob Charles, AKA Rio Kikaru submitted for the MinorHack challenge at September 13th 2008 at 7:00 pm UTC.

// gravitypanic.cpp listing begins:

```
#include <allegro.h>

#define MODE GFX_AUTODETECT_WINDOWED
#define WIDTH 640
#define HEIGHT 480
#define COLOR_DEPTH 32

#define BLACK makecol(0, 0, 0)
#define WHITE makecol(255, 255, 255)

#define LOOP_TIME 16

class platform
{
public:
int x, y, wid, hei;
};

//standard variables
BITMAP *buffer;
long start_time;
long long timer = 0;
bool end_game = false;
bool game_over = false;

platform land[40];
platform coin[30];
int n_lands, n_coins;

//player
int x, y, vx = 0, vy = 0, jump = 2, score = 0;
bool jumped;

//gravity
int x_grav = 0, y_grav = 10;

//update the timer
void update_timer(){timer++;}

//quit the game
void quit(){end_game = true;}

void random_area()
{
    n_lands = (rand()%20)+10;
    n_coins = (rand()%10)+10;
    for (int i = 0; i < n_lands; i++)
    {
  land[i].x = 32+rand()%(WIDTH-64);
  land[i].y = 32+rand()%(HEIGHT-64);
  land[i].wid = 24+rand()%24;
  land[i].hei = 24+rand()%24;
    }
    for (int i = 0; i < n_coins; i++)
    {
  coin[i].x = 32+rand()%(WIDTH-64);
  coin[i].y = 32+rand()%(HEIGHT-64);
  coin[i].wid = 24;
  coin[i].hei = 24;
    }
    int xi = rand()%n_lands;
```

// Listing continued on next page…

// Listing continued from previous page



```
    x = land[xi].x+land[xi].wid/2;
    y = land[xi].y;
    score = 0;
}

//main function
void main()
{
    //setup allegro
    allegro_init();
    install_keyboard();
    install_mouse();
    install_timer();
    set_color_depth(COLOR_DEPTH);
    set_gfx_mode(MODE, WIDTH, HEIGHT, 0, 0);

    //setup the timer
    install_int_ex(update_timer, MSEC_TO_TIMER(1));

    //activate the close button
    set_close_button_callback(quit);

    //seed the random number
    srand(time(NULL));

    //initialize the buffer
    buffer = create_bitmap(WIDTH, HEIGHT);

    textprintf_centre_ex(screen, font, WIDTH/2, HEIGHT/2-30, WHITE, -1,
      "~Gravity Panic~");
    textprintf_centre_ex(screen, font, WIDTH/2, HEIGHT/2-10, WHITE, -1,
      "Move with the arrow keys");
    textprintf_centre_ex(screen, font, WIDTH/2, HEIGHT/2+5, WHITE, -1,
      "Collect coins to get points.");
    textprintf_centre_ex(screen, font, WIDTH/2, HEIGHT/2+20, WHITE, -1,
      "Use WSAD to control the direction of gravity.");
    textprintf_centre_ex(screen, font, WIDTH/2, HEIGHT/2+35, WHITE, -1,
      "You can't control gravity without twice between touching the ground.");
    textprintf_centre_ex(screen, font, WIDTH/2, HEIGHT/2+50, WHITE, -1,
      "Hit 'R' to generate a new level.");

    while ((!key[KEY_ENTER])&&(!key[KEY_ESC])&&(!end_game))rest(2);

    random_area();

    //main loop
    while (!end_game)
    {
  //log the start time
  start_time = timer;

  //press esc to quit
  if (key[KEY_ESC])
    end_game = true;

  /*game goes here*/
  //gravity control
  if ((key[KEY_W])&&(jump > 0)&&(!jumped))
    {x_grav = 0; y_grav = -10; jump=0; jumped = true;}
  if ((key[KEY_S])&&(jump > 0)&&(!jumped))
    {x_grav = 0; y_grav = 10; jump=0; jumped = true;}
  if ((key[KEY_A])&&(jump > 0)&&(!jumped))
    {x_grav = -10; y_grav = 0; jump=0; jumped = true;}
```

// Listing continued on next page…

// Listing continued from previous page

```
  if ((key[KEY_D])&&(jump > 0)&&(!jumped))
    {x_grav = 10; y_grav = 0; jump=0; jumped = true;}

  if (vx < x_grav) vx++;
  if (vx > x_grav) vx--;
  if (vy < y_grav) vy++;
  if (vy > y_grav) vy--;
  if (x_grav == 0) vx = 0;
  if (y_grav == 0) vy = 0;

  if ((key[KEY_LEFT])&&(x_grav == 0)) vx = -3;
  if ((key[KEY_LEFT])&&(x_grav == 10)&&(jump > 1)&&(!jumped))
    {vx = -10; jump--; jumped = true;}
  if ((!key[KEY_LEFT])&&(x_grav == 10))
    {jumped = false;}

  if ((key[KEY_RIGHT])&&(x_grav == 0)) vx = 3;
  if ((key[KEY_RIGHT])&&(x_grav == -10)&&(jump > 1)&&(!jumped))
    {vx = 10; jump--; jumped = true;}
  if ((!key[KEY_RIGHT])&&(x_grav == -10)) {jumped = false;}

  if ((key[KEY_UP])&&(y_grav == 0)) vy = -3;
  if ((key[KEY_UP])&&(y_grav == 10)&&(jump > 1)&&(!jumped))
    {vy = -10; jump--; jumped = true;}
  if ((!key[KEY_UP])&&(y_grav == 10)) {jumped = false;}

  if ((key[KEY_DOWN])&&(y_grav == 0)) vy = 3;
  if ((key[KEY_DOWN])&&(y_grav == -10)&&(jump > 1)&&(!jumped))
    {vy = 10; jump--; jumped = true;}
  if ((!key[KEY_DOWN])&&(y_grav == -10)) {jumped = false;}

  if (key[KEY_R])random_area();

  //collision
  for (int i = 0; i < n_lands; i++)
  {
    if ((x > land[i].x) && (x < land[i].x+land[i].wid)
      && (y > land[i].y) && (y < land[i].y+land[i].hei))
    {
      if (((x_grav < 0)&&(vx < 0))||((x_grav > 0)&&(vx > 0))) vx = 0;
      if (((y_grav < 0)&&(vy < 0))||((y_grav > 0)&&(vy > 0))) vy = 0;
      jump = 2; jumped = false;
    }
  }
  for (int i = 0; i < n_coins; i++)
  {
    if ((x > coin[i].x) && (x < coin[i].x+coin[i].wid) && (y > coin[i].y)
      && (y < coin[i].y+coin[i].hei))
    {
      coin[i] = coin[n_coins-1];
      n_coins--;
      score++;
    }
  }
  x += vx;
  y += vy;

  if ((x < 0)&&(x_grav < 0)) game_over = true;
  if ((x > WIDTH)&&(x_grav > 0)) game_over = true;
  if ((y < 0)&&(y_grav < 0)) game_over = true;
  if ((y > HEIGHT)&&(y_grav > 0)) game_over = true;

  //game over
```

// Listing continued on next page…

// Listing continued from previous page

```
  while ((game_over)&&(!end_game))
  {
  textprintf_centre_ex(screen, font, WIDTH/4, HEIGHT/4, WHITE, -1, "Game Over");
  textprintf_centre_ex(screen, font, WIDTH/4, HEIGHT*3/4, WHITE, -1,
    "Game Over");
  textprintf_centre_ex(screen, font, WIDTH*3/4, HEIGHT/4, WHITE, -1,
    "Game Over");
  textprintf_centre_ex(screen, font, WIDTH*3/4, HEIGHT*3/4, WHITE, -1,
    "Game Over");
  //press esc to quit
  if (key[KEY_ESC])
    end_game = true;
  rest(1);
  }

  /*drawing goes here*/
  circlefill(buffer, x, y, 5, WHITE);
  for (int i = 0; i < n_lands; i++)
  {rect(buffer, land[i].x, land[i].y, land[i].x+land[i].wid, land[i].y+land
[i].hei, WHITE);}
  for (int i = 0; i < n_coins; i++)
  {circle(buffer, coin[i].x+12, coin[i].y+12, 5, makecol(255, 255, 0));}

  textprintf_centre_ex(buffer, font, WIDTH/2, HEIGHT/2, WHITE, -1, "%i", score);

  //draw to the screen
  blit(buffer, screen, 0, 0, 0, 0, WIDTH, HEIGHT);

  //clear the buffer
  clear_to_color(buffer, BLACK);

  //wait
  while (timer < start_time + LOOP_TIME)
  {rest(1);}
    }
    //clean up
    remove_int(update_timer);
    destroy_bitmap(buffer);
    return;
}
END_OF_MAIN();
```