Baddie

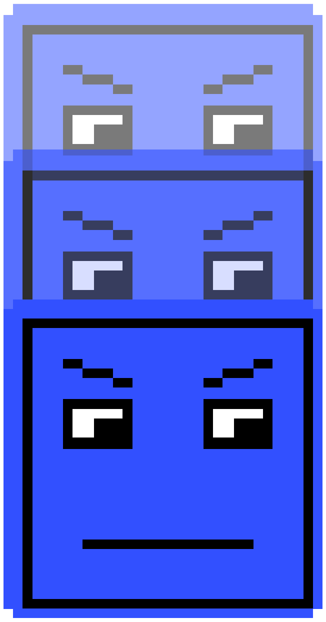
How’s your trigger finger? Hope it’s fast because we’ve got baddies.

Baddie is a twitch shooter where speed and accuracy count. The game encourages you to make the most of each round when you’re fighting for a high score.

The mouse controls your crosshair. Left mouse button shoots, right mouse button reloads. Each reload costs 250 points. Gain points by shooting baddies and coins. If a baddie makes it to the bottom of the screen you lose a hit point (HP). When your HP is zero the game is over.

Baddie is by Jakub Wasilewski, submitted for the MinorHack challenge at September 3rd 2006.

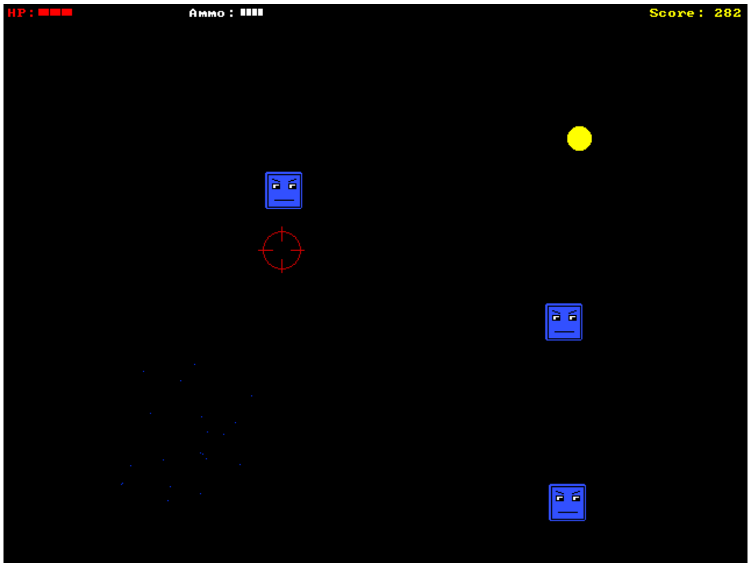
// baddie.cpp listing begins:



```
#include <cstdlib>
#include <cmath>
#include <ctime>
#include <allegro.h>
#include <string>
#include <list>
#include <vector>
using namespace std;
/******************************************************/
BITMAP *buffer, *crosshair, *baddie;
int score = 0, hp = 4, ammo = 6;
float level;
/******************************************************/
static volatile int ticks = 0;
void incTicks() {
  ticks++;
}
void resetTicks() {
  ticks = 0;
}
/******************************************************/
void initRandom() {
  std::srand(time(NULL));
}
/******************************************************/
float randomFloat(float min, float max) {
  return min + ((float)std::rand() / (float)RAND_MAX) * (max - min);
}
/************************************************************/
long randomInt(long min, long max) {
  return rand() % (max - min + 1) + min;
}
/**********************************************************/
void init() {
  allegro_init();
  set_color_depth(desktop_color_depth());
  set_gfx_mode(GFX_AUTODETECT_WINDOWED, 640, 480, 0, 0);
  install_keyboard(); install_timer(); install_mouse();
  install_int_ex(incTicks, BPS_TO_TIMER(60));
  initRandom();
  // buffer
  buffer = create_bitmap(640, 480);
  // crosshair
  crosshair = create_bitmap(40, 40);
  clear_to_color(crosshair, makecol(255, 0, 255));
  circle(crosshair, 20, 20, 16, makecol(200, 0, 0));
  line(crosshair, 20, 0, 20, 12, makecol(200, 0, 0));
  line(crosshair, 20, 40, 20, 28, makecol(200, 0, 0));
  line(crosshair, 0, 20, 12, 20, makecol(200, 0, 0));
  line(crosshair, 40, 20, 28, 20, makecol(200, 0, 0));
  // baddie
  baddie = create_bitmap(32, 32);
  clear_to_color(baddie, makecol(255, 0, 255));
  rectfill(baddie, 1, 0, 30, 31, makecol(50, 80, 255));
  rectfill(baddie, 0, 1, 31, 30, makecol(50, 80, 255));
  rect(baddie, 2, 2, 30, 30, makecol(0, 0, 0));
  rectfill(baddie, 6, 10, 12, 14, makecol(255, 255, 255));
  rectfill(baddie, 20, 10, 26, 14, makecol(255, 255, 255));
  rect(baddie, 6, 10, 12, 14, makecol(0, 0, 0));
  rect(baddie, 20, 10, 26, 14, makecol(0, 0, 0));
  rectfill(baddie, 9, 12, 11, 14, makecol(0, 0, 0));
```

// Listing continued on next page…

// Listing continued from previous page



```
  rectfill(baddie, 23, 12, 25, 14, makecol(0, 0, 0));
  line(baddie, 6, 6, 12, 8, makecol(0, 0, 0));
  line(baddie, 20, 8, 26, 6, makecol(0, 0, 0));
  line(baddie, 8, 24, 24, 24, makecol(0, 0, 0));
}
/****************************************/
void destroy() {
}
/*******************************************/
class Thing {
  public:
  virtual bool    isAlive() {return true;};
  virtual void    update(){};
  virtual void  draw(){};
  virtual void    shootAt(int x, int y){};
};
list<Thing*> world;
class Gun : public Thing {
  int x, y;

  public:
  void update() { x = mouse_x; y = mouse_y; }
  void draw() { draw_sprite(buffer, crosshair, x - 20, y - 20); }
};
class Particle : public Thing {
  public:
  int color;
  float x, y;
  float vx, vy;
  Particle(int x, int y, float vx, float vy, int color)
    : x(x), y(y), vx(vx), vy(vy), color(color) {}
  bool isAlive() {
    return y < 480;
  }
  void update() {
    vy += 0.2;
    x += vx;
    y += vy;
  }
  void draw() {
    putpixel(buffer, (int)x, int(y), color);
  }
};
class Bonus : public Thing {
  public:
  float x, y;
  float vy;
  bool destroyed;
  public:
  Bonus(int x, float vy) : x(x), vy(vy) { y = -20; destroyed = false; }
  bool isAlive() { return (!destroyed); }
  void draw() { circlefill(buffer, (int)x, int(y), 10, makecol(255, 255, 0)); }
  void update() {
    y += vy;
    if (y >= 490) destroyed = true;
  }

  void shootAt(int sx, int sy) {
    if (((sx - x) * (sx - x) + (sy - y) * (sy - y)) < 100) {
      score += (int)(level * 100);
      destroyed = true;
      for(int i = 0; i < 20; i++) {
        world.push_back(new Particle((int)x + randomInt(-5, 5)
```

// Listing continued on next page…

// Listing continued from previous page

```
        , (int)y + randomInt(-5, 5), randomFloat(-2, 2)
        , randomFloat(-2, 2), makecol(255, 200, 0)));
      }
    }
  }
};


class Baddie : public Thing {
  public:

  float x, y;
  float vy;
  bool destroyed;

  public:

  Baddie(int x, float vy) : x(x), vy(vy) {
    y = -20;
    destroyed = false;
  }
  bool isAlive() { return (!destroyed); }
  void draw() { draw_sprite(buffer, baddie, (int)x - 16, (int)y - 16); }
  void update() {
    y += vy;
    if (y >= 490) { hp--; destroyed = true; }
  }
  void shootAt(int sx, int sy) {
    if ((std::abs(sx - x) < 15) && (std::abs(sy - y) < 15)) {
      score += (int)(level * 25);
      destroyed = true;
      for(int i = 0; i < 20; i++) {
        world.push_back(new Particle((int)x + randomInt(-5, 5)
          , (int)y + randomInt(-5, 5), randomFloat(-2, 2)
          , randomFloat(-2, 2), makecol(0, 50, 255)));
      }
    }
  }
};
class Scoreboard : public Thing {
  public:
  void draw() {
    textprintf_ex(buffer, font, 4, 4, makecol(255, 0, 0), -1, "HP:");
    for (int i = 0; i < hp; i++)
      rectfill(buffer, 30 + i * 10, 4, 30 + i * 10 + 8, 9, makecol(255, 0, 0));
    textprintf_ex(buffer, font, 160, 4, makecol(255, 255, 255), -1, "Ammo:");
    for (int i = 0; i < ammo; i++)
      rectfill(buffer,204 + i * 5, 4,204 + i * 5 + 3, 9, makecol(255, 255, 255));
    textprintf_right_ex(buffer, font, 636, 4, makecol(255, 255, 0), -1
      , "Score: %d", score);
  }
};
/*********************************************************/
int main() {
  init();
  resetTicks();
  bool end = false;
  list<Thing*>::iterator it;
  Gun *gun = new Gun();
  Scoreboard *sb = new Scoreboard();
  level = 4.0;
  int prev_button = 0;
  while(!end) {
```

// Listing continued on next page…

// Listing continued from previous page

```
    if (ticks > 0) {
      while(ticks-- > 0) {
        if (randomFloat(0, 48 - level) < 1.0) {
          if (randomInt(0, 15) == 0)
            world.push_back(new Bonus(randomInt(20, 620)
              , level * randomFloat(0.99, 1.05)));
          else
            world.push_back(new Baddie(randomInt(20, 620)
              , level * randomFloat(0.99, 1.05)));
        }
        for(it = world.begin(); it != world.end(); it++) (*it)->update();
        for(it = world.begin(); it != world.end();) {
          if (!(*it)->isAlive()) {
            delete (*it);
            it = world.erase(it);
          }
          else {
            it++;
          }
        }
        gun->update();\
        if (key[KEY_ESC]) {
          end = true;
        }
        end |= (hp <= 0);
        level += 0.0012;
        if ((mouse_b == 1) && (mouse_b != prev_button) && (ammo > 0)) {
          ammo--;
          for(it = world.begin(); it != world.end(); it++)
            (*it)->shootAt(mouse_x, mouse_y);
        }
        if ((mouse_b == 2) && (mouse_b != prev_button)) {
          ammo = 6;
          score -= 250;
        }
        prev_button = mouse_b;
      }
    }
    clear_bitmap(buffer);
    for(it = world.begin(); it != world.end(); it++) (*it)->draw();
    gun->draw(); sb->draw();
    blit(buffer, screen, 0, 0, 0, 0, 640, 480);
  }
  clear_bitmap(screen);
  textprintf_centre_ex(screen, font, 320, 220, makecol(255, 0, 0), -1
    , "G A M E   O V E R");
  textprintf_centre_ex(screen, font, 320, 240, makecol(255, 255, 255), -1
    , "Final score: %d", score);
  textprintf_centre_ex(screen, font, 320, 260, makecol(255, 255, 0), -1
    , "Press both mouse buttons to exit");
  while (mouse_b);
  while (mouse_b != 3);
  destroy();
}
END_OF_MAIN()
```