University of Reading

Department of Computer Science

CS1PC20 - Programming in C++

Coursework 4 - Modifying a Game in C/C++

Ruben j. Lopes

Module Code:	CS1PC20
Report Title:	Coursework 4
Student Number:	30021591
Date:	28th March 2022
hrs spent:	25hr
Git Repository:	`https://csgitlab.reading.ac.uk/il021591/cs1pc20_Spring_Project.git`
Lecturer:	Dr Ashraf Mahmud
Weighting of the Assignment:	30%
Assignment evaluation:	Impoved understanding and confidance programming in C. Learned the impartance of commenting and documentation for mantainance.

Contents

Declaration

I, Ruben Lopes, of the Department of Computer Science, University of Reading, corm that all the sentences, gures, tables, equations, code snippets, artworks, and illus- trations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

— Ruben Lopes
February 1, 2021

1 Introduction

The goal of this project was to demonstrate our capacity to analyse, create, and implement current C/C++ code to add new features. The SLD2 library was included in the skeleton code for the basic game which we had to utilize to run the graphical components of the game. The Theme I used throughout was a moon-based space game. Making the player in this case a human astronaut. The main objective for the player in this game is to collect all the neon blue orbs on each map. The player will have to navigate across the maps using platforms and blocks and will have to avoid taking damage from the aliens.

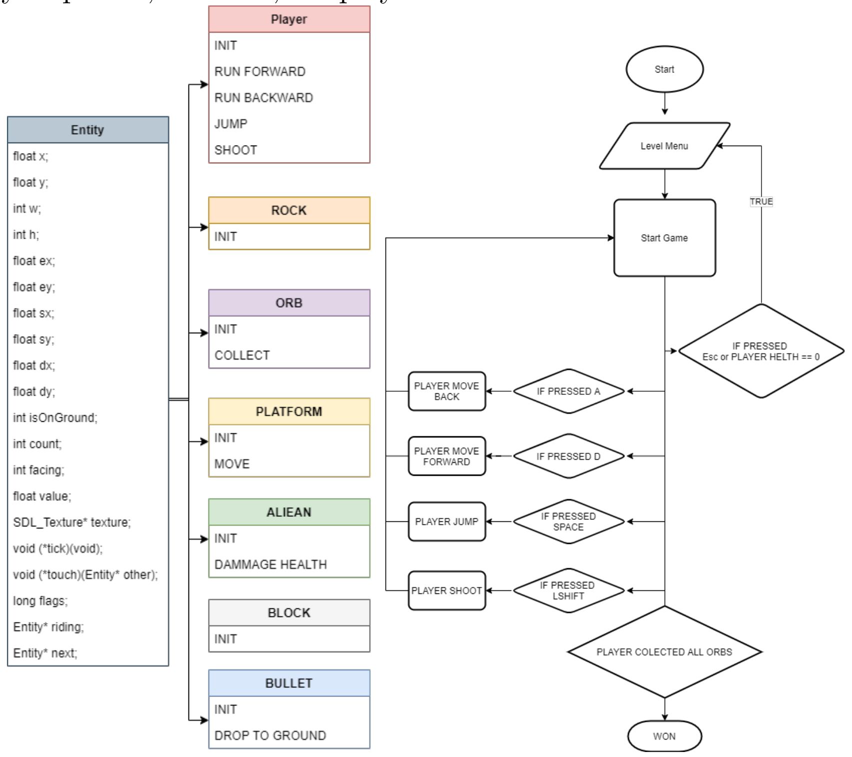
I have implemented about five features, four new maps and have created twenty-two textures. The features that I have implemented were the ability for the player to choose to do a different map or level. Additionally, a menu screen is displayed at the start of the game to allow the user to choose one of the five maps/levels made available to them. Moving Aliens that decrees the players health or kills the player when the character interacts with it was also implemented and the player was given the ability to shoot when the 'shift' button was held, this allowed the player to kill the Alien. In addition to functional change, I have also changed the texture of the blocks, character, background, and platform to fit the theme of the game.

The Programming style used by me is mainly imperative as C was predominately used to implement all the new features. Although for components such as entities structs were used to store objects such as enemies and player information so it can be more accessible to other functions. The public functions in the skeleton code and some of the ones I implemented were stored in the commons.h file.

In this report, I will go through the development decisions and design choices I made for these features I implemented. This report will consist of four additional components. In design I will be focusing on the UI and textures and the objects and its features. In the implementation and development section I will go into more detail about each feature and describe the development process of each feature more clearly, additionally, I will Justify my changes compared to keeping the original design. And finally, describe improvements or alternative features I could have used in the conclusions. The difference of code will also be available at the end of this document, comparing the original skeleton code provided to the code that I wrote.

2 Design

The new maps or levels encourage gamers to explore the game in a variety of terrains. The addition of aliens made the game more difficult, and this feature also permitted the addition of laser weapons. The alien's movement is implemented using the same mechanism as the platforms provided in the skeleton code. With a few changes, the shooting functionality was also easily implemented utilizing the pre-existing struct. Because having infinite weapon ammo would make the game easy, I set a restriction to how much it can be used by utilizing a counter and only allowing one hundred bullets when the player spawns, however, the player earns more bullets when orbs are gathered.

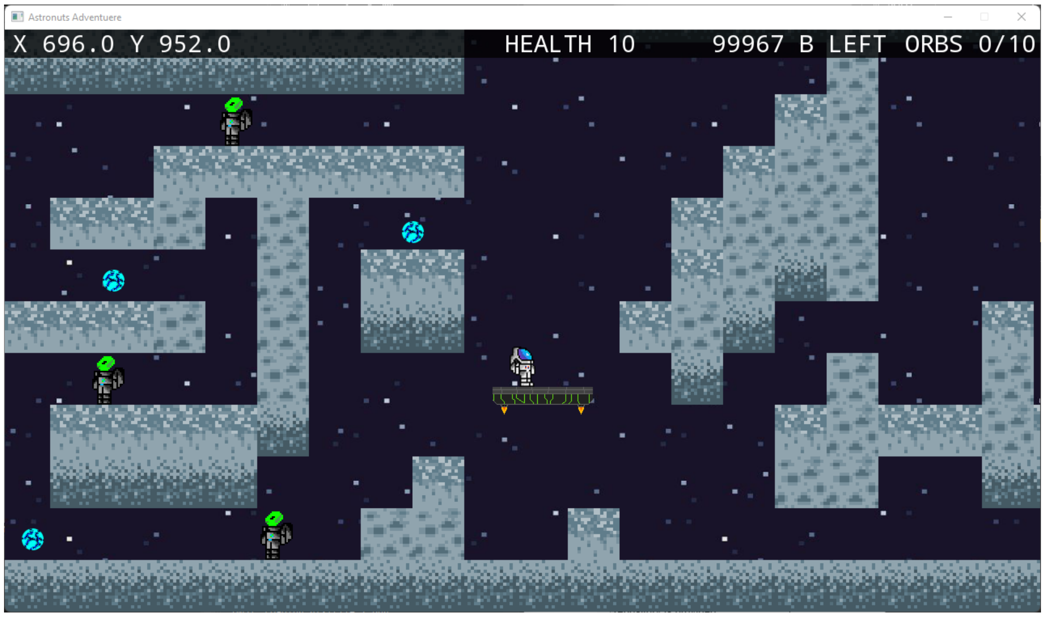


The controls to the player are listened for in the game loop once the game starts. To interact with the game, the user can then move, shoot, or jump using W A S D Right Shift or spacebar. If the player gathers all of the orbs on the map, the game finishes, and the level is completed. If the player touches the alien, it loses health, and if it falls below zero, the game terminates, and the player is returned to the menu. This is visualized using the flowchart below.

3 Implementation and development

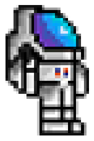
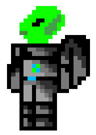
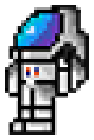
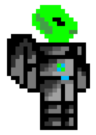
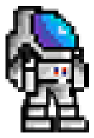
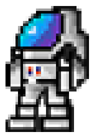
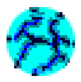
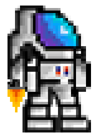
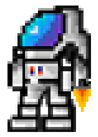
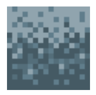
Since the core of the programming was provided to us in the skeleton code to add new features, I used the pre-existing code and designed and built the additional features based on it. Firstly, I developed separate files called attack.c, emeny.c, and menu.c for the three primary new features. I made some other adjustments after adding it to the commons header file and the struct file.

3.1 New Maps or Levels



We were provided a single map in the skeleton code. These were set up using a 40 by 60 number matrix ranging from 1 to 7. The initmap function then read this by calling the io.c file. The map was created, and the entities from the entX.dat folder were placed depending on their coordinates. These files served as the foundation for the additional four maps I produced. Similarly, to the maps, I had to create four more entity files to add the entities to the new maps. The Alien entity was also updated to be able to be added using this method. during the map making process I found it difficult to determine the correct coordinates while creating the map, so I used the drawText method to print out the players' x and y coordinates. This, however, was solely for development and testing and will be deleted in the final version.

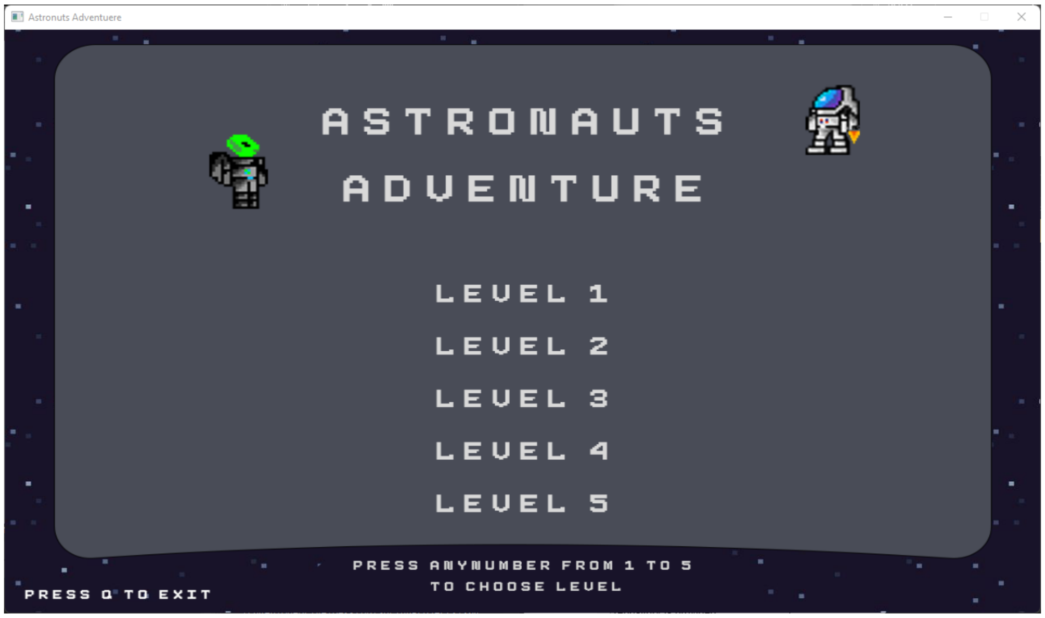
3.2 Game Textures and theme

Astronaut Right Still		Alien Right Still	
Astronaut Left Still		Alien Left Still	
Astronaut Right Walk		Rock 1	
Astronaut Left Walk		Rock 2	
		Lazar Shot	
		Orb	
Astronaut Right Jump		Moon Topsoil	
Astronaut Left Jump		Moon Rock	
		Moon Bedrock	

The controls to the player are listened for in the game loop once the game starts. To interact with the game, the user can then move, shoot, or jump using W A S D Right Shift or spacebar. If the player gathers all of the orbs on the map, the game finishes, and the level is completed. If the player touches the alien, it loses health, and if it falls below zero, the game terminates, and the player is returned to the menu. This is visualized using the flowchart below.

The game, as indicated in the introduction, is based on a shooting game with a moon theme. I choose pixel art as the art style for this game. All of the textures in the game were created in Photoshop using a certain colour palette and aspect ratio. The player includes six distinct texture pictures for the left and right positions of the three functions it may perform: stand, walk, and leap. Similarly, the alien had textures on both the left and right sides. The platform was designed to seem like an industrial Si-Fi spaceship platform. Three tiles were made with topsoil, dirt, and bedrock designs to make the terrain look like the moon. Rocks were added for astatic, and the pizza was also changed to a blue orb to fit the theme.

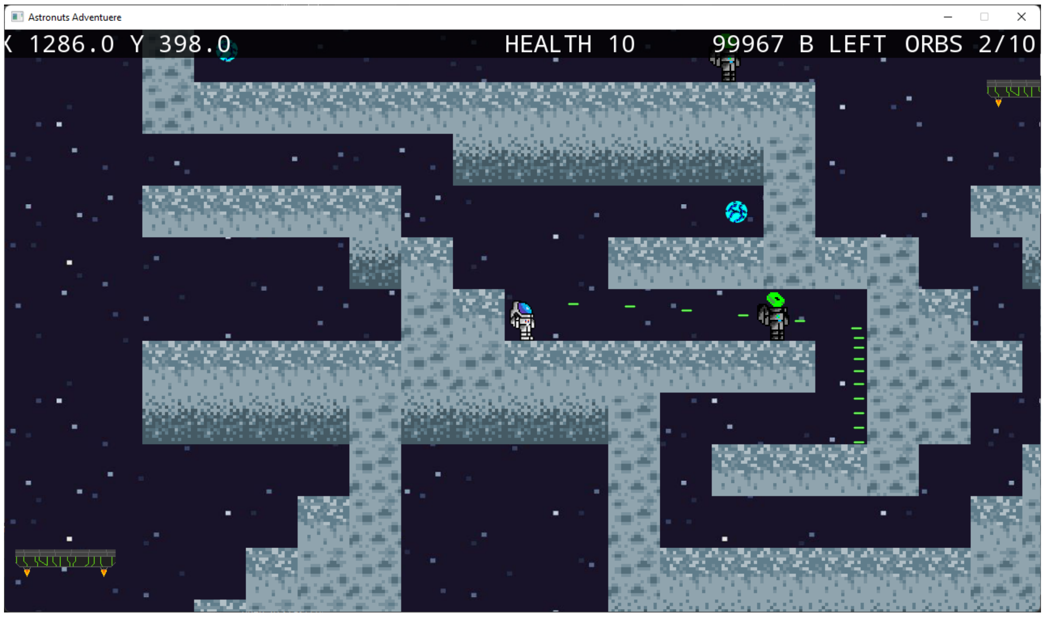
3.3 Game Menu



The game menu is comprised of a basic backdrop image created in Photoshop. In the same way as the game does, this loops until any number from 1 to 5 is pressed to begin the level. Alternatively, Q could be pressed to exit the application. Players may also return to this menu by pressing esc during the game, although their progress will be lost.

```

#include "common.h"

void initMenu(void)
{
  SDL_ShowCursor(1);
  long then;
  then = SDL_GetTicks();
  //SDL_SetRenderDrawColor(app.renderer, 200, 53, 53, 235);

  SDL_Texture* menuImg = IMG_LoadTexture(app.renderer, "gfx/menu.png");

  int exit = 0;

  while (!exit)
  {
    SDL_RenderCopy(app.renderer, menuImg, NULL, NULL);
    SDL_RenderPresent(app.renderer);

    doInput();

    if (app.keyboard[SDL_SCANCODE_1])
    {
      level = 1;
      exit = 1;
    }
    if (app.keyboard[SDL_SCANCODE_2])
    {
      level = 2;
      exit = 1;
    }
    if (app.keyboard[SDL_SCANCODE_3])
    {
      level = 3;
      exit = 1;
    }
    if (app.keyboard[SDL_SCANCODE_4])
    {
      level = 4;
      exit = 1;
    }
    if (app.keyboard[SDL_SCANCODE_5])
    {
      level = 5;
      exit = 1;

    }
    if (app.keyboard[SDL_SCANCODE_Q])
    {
      cleanup();
    }
    presentScene();
  }
  initStage();

}
```

3.4 Player Attacks (Shooting)

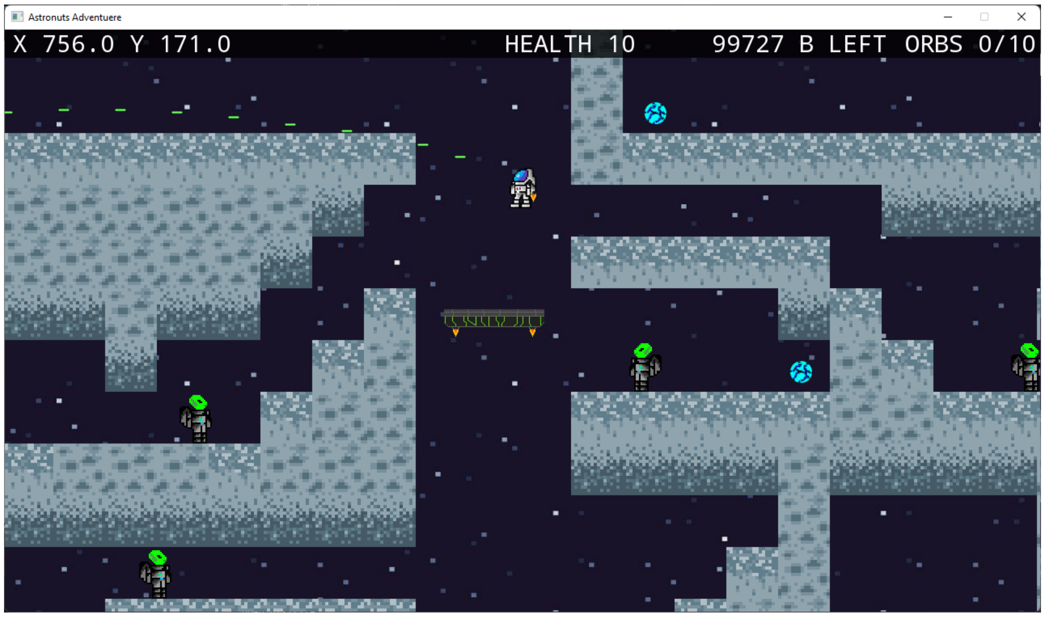


The shooting functionality utilises the entity struct to create bullets. When a collision is detected, these bullets activate the touch function. If it collides with an entity, it will cause harm, and if it collides with the ground, it will disappear. The bullet's velocity is stored in the struct's dy variable, which uses the physics of all the other entities previously supplied and is inverted according on the player's location, for example, if the player fires towards the left, the bullet is shot from the left. The bullets are also restricted to one hundred every game and will − 1 for each shot until they reach 0, and the player will be unable to fire the laser gun unless an orb is obtained, which grants the player +5 bullets.

```

#include "common.h"
static void tick(void);
static void touch(Entity* other);

static SDL_Texture* bulletTex[1];

void initBullet(void)
{
  bullet = malloc(sizeof(Entity));
  memset(bullet, 0, sizeof(Entity));
  stage.entityTail->next = bullet;
```

```c
  stage.entityTail = bullet;


  bullet->health = 1;
  bullet->x = player->x;
  bullet->y = player->y;
  bullet->touch = touch;
  bullet->tick = tick;
  bullet->dx = 70;

  if (stage.bulletCount <= 0) { return; }
  else{stage.bulletCount -= 1;}

  if (player->facing == RIGHT)
  {
    bullet->dx = 70;
  }
  else if (player->facing == LEFT)
  {
    bullet->dx = -70;
  }

  bulletTex[0] = loadTexture("gfx/shot.png");
  bullet->texture = bulletTex[0];

  SDL_QueryTexture(bullet->texture, NULL, NULL, &bullet->w, &bullet->h);
}

static void touch(Entity* other)
{
  if (other == self)
  {
    self->health -= 1;
    player->value += 1;

  }
}

static void tick(void)
{
  if (self->isOnGround)
  {
    self->health = 0;
  }
}
```

3.5 Enemy (Aliens)



The aliens were designed to add a level of difficulty to the game. Alien movement from sx to ex is similar to platforms and may be set in the etntery.dat file. If the player fires at these aliens, they will receive damage, and if the player touches them, they will kill the player instantaneously. The aliens have two textures, which are kept in an array of SDL texture. It likewise made use of the same struct as the other entities.

```

#include "common.h"

static void tick(void);
static void touch(Entity* other);
SDL_Texture* texture[1];

void initEnemy(char* line)
{
  Enemy* e;

  e = malloc(sizeof(Enemy));
  memset(e, 0, sizeof(Enemy));
  stage.entityTail->next = e;
  stage.entityTail = e;

  sscanf(line, "%*s %f %f %f %f", &e->sx, &e->sy, &e->ex, &e->ey);

  e->health = 100;

  e->x = e->sx;
  e->y = e->sy;

  e->tick = tick;
  e->touch = touch;
  e->flags = EF_PUSH;

  texture[0] = loadTexture("gfx/alienF1.png");
  texture[1] = loadTexture("gfx/alienB1.png");
  e->texture = texture[0];

  SDL_QueryTexture(e->texture, NULL, NULL, &e->w, &e->h);
}

static void tick(void)
{
  if (abs(self->x - self->sx) < PLATFORM_SPEED)
  {
    calcSlope(self->ex, self->ey, self->x, self->y, &self->dx, &self->dy);

    self->dx *= PLATFORM_SPEED;
    self->dy *= PLATFORM_SPEED;
    self->texture = texture[0];
  }
  if (abs(self->x - self->ex) < PLATFORM_SPEED)
  {
    calcSlope(self->sx, self->sy, self->x, self->y, &self->dx, &self->dy);

    self->dx *= PLATFORM_SPEED;
    self->dy *= PLATFORM_SPEED;
    self->texture = texture[1];
  }
}

static void touch(Enemy* other)
{
  if (other == player)
  {
    other->health -= 1;
  }
}
```

4 Conclusions

In conclusion this coursework project helped me understand the importance of reliability of code and the steps to maintain and update repositories. I Also helped me to improve my understanding as well as confidence C and improved my problem-solving skill. Besides programming, designing textures was also something I learned. If I were to do this project again, I would plan out a game more thoroughly and use a software development life cycle such as sprint and a Kanban board to manage the project. Furthermore, I could have more comments and audited the progress better to reduce setbacks in managing a project of a multiple files.

I have recognized that I could improve my productivity if I learn to use Git and a testing framework as I struggled to maintain my code and often failed to remember where I last modified code. And this could also aid in have a more inadept testing of features as I could test and push iteratively.

5 Git repository diff

```
diff --git a/src/attack.c b/src/attack.c
new file mode 100644
index 0000000..5b0ceee
--- /dev/null
+++ b/src/attack.c
@@ -0,0 +1,57 @@
+
+#include "common.h"
+static void tick(void);
+static void touch(Entity* other);
+
+static SDL_Texture* bulletTex[1];
+
+void initBullet(void)
+{
+ bullet = malloc(sizeof(Entity));
+ memset(bullet, 0, sizeof(Entity));
+ stage.entityTail->next = bullet;
+ stage.entityTail = bullet;
+
+
+ bullet->health = 1;
+ bullet->x = player->x;
+ bullet->y = player->y;
+ bullet->touch = touch;
+ bullet->tick = tick;
+ bullet->dx = 70;
+
+ if (stage.bulletCount <= 0) { return; }
+ else{stage.bulletCount -= 1;}
+
+ if (player->facing == RIGHT)
+ {
+    bullet->dx = 70;
+ }
+ else if (player->facing == LEFT)
+ {
+    bullet->dx = -70;
+ }
+
+ bulletTex[0] = loadTexture("gfx/shot.png");
+ bullet->texture = bulletTex[0];
+
+ SDL_QueryTexture(bullet->texture, NULL, NULL, &bullet->w, &bullet->h);
+}
+
+static void touch(Entity* other)
+{
+ if (other == self)
+ {
+    self->health -= 1;
+    player->value += 1;
+
+ }
+}
+
+static void tick(void)
+{
+ if (self->isOnGround)
+ {
+    self->health = 0;
+ }
+}
\ No newline at end of file
diff --git a/src/attacks.c b/src/attacks.c
new file mode 100644
index 0000000..73f7384
--- /dev/null
+++ b/src/attacks.c
@@ -0,0 +1,17 @@
+
+#include "common.h"
+
+static SDL_Texture* bullet;
+
+
+void initBullet(void)
+{
+ bullet = malloc(sizeof(Entity));
+ memset(bullet, 0, sizeof(bullet));
+
+ bullet = loadTexture("gfx/shot.png");
+
+ bullet->texture = bullet[0];
+
+ SDL_QueryTexture(bullet->texture, NULL, NULL, &bullet->w, &bullet->h);
+}
\ No newline at end of file
```

```
diff --git a/src/block.c b/src/block.c
index 6271389..b2bd2cc 100644
--- a/src/block.c
+++ b/src/block.c
@@ -37,3 +37,41 @@ void initBlock(char *line)
  SDL_QueryTexture(e->texture, NULL, NULL, &e->w, &e->h);
  e->flags = EF_SOLID+EF_WEIGHTLESS;
 }
+
+void initRock1(char* line)
+{
+ Entity* e;
+
+ e = malloc(sizeof(Entity));
+ memset(e, 0, sizeof(Entity));
+ stage.entityTail->next = e;
+ stage.entityTail = e;
+
+ sscanf(line, "%*s %f %f", &e->x, &e->y);
+
+ e->health = 100;
+
+ e->texture = loadTexture("gfx/rock1.png");
+ SDL_QueryTexture(e->texture, NULL, NULL, &e->w, &e->h);
+ //e->flags = EF_SOLID + EF_WEIGHTLESS;
+ e->flags = EF_PUSH;
+}
+
+void initRock2(char* line)
+{
+ Entity* e;
+
+ e = malloc(sizeof(Entity));
+ memset(e, 0, sizeof(Entity));
+ stage.entityTail->next = e;
+ stage.entityTail = e;
+
+ sscanf(line, "%*s %f %f", &e->x, &e->y);
+
+ e->health = 100;
+
+ e->texture = loadTexture("gfx/rock2.png");
+ SDL_QueryTexture(e->texture, NULL, NULL, &e->w, &e->h);
+ //e->flags = EF_SOLID + EF_WEIGHTLESS;
+ e->flags = EF_NONE + EF_PUSH;
```

```
+}
diff --git a/src/common.h b/src/common.h
index 27923f5..8d57c2c 100644
--- a/src/common.h
+++ b/src/common.h
@@ -21,9 +21,17 @@ extern "C" {

 char * global_dir;
 App app;
-Entity *player;
+Entity* player;
 Stage stage;
 Entity *self;
+Entity* bullet;
+int* level;
+
+
+void initMenu(void);
+void initBullet(void);
+void initEnemy(char* line);
+void levelCompleted(int level);

 char *readFile(const char *filename);
 int collision(int x1, int y1, int w1, int h1, int x2, int y2, int w2, int h2);
@@ -45,7 +53,7 @@ void initEntities(void);
 void initFonts(void);
 void initGame(void);
 void initMap(void);
-void initPizza(char *line);
+void initOrb(char *line);
 void initPlatform(char *line);
 void initPlayer(void);
 void initSDL(void);
@@ -57,6 +65,7 @@ void playSound(int id, int channel);
 void prepareScene(void);
 void presentScene(void);

+
 #ifdef __cplusplus
 }
 #endif
diff --git a/src/defs.h b/src/defs.h
index 1135ab7..67a23cd 100644
--- a/src/defs.h
+++ b/src/defs.h
```

```
@@ -54,6 +54,9 @@ Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
 #define GLYPH_WIDTH  18
 #define GLYPH_HEIGHT 29
 
+#define RIGHT 1
+#define LEFT -1
+
 enum
 {
  TEXT_LEFT,
@@ -64,13 +67,13 @@ enum
 enum
 {
  SND_JUMP,
- SND_PIZZA,
- SND_PIZZA_DONE,
+ SND_ORB,
+ SND_ORB_DONE,
  SND_MAX
 };
 
 enum
 {
  CH_PLAYER,
- CH_PIZZA
+ CH_ORB
 };
diff --git a/src/enemy.c b/src/enemy.c
new file mode 100644
index 0000000..6e3b78c
--- /dev/null
+++ b/src/enemy.c
@@ -0,0 +1,65 @@
+
+#include "common.h"
+
+static void tick(void);
+static void touch(Entity* other);
+SDL_Texture* texture[1];
+
+
+void initEnemy(char* line)
+{
+ Enemy* e;
+
+ e = malloc(sizeof(Enemy));
+ memset(e, 0, sizeof(Enemy));
+ stage.entityTail->next = e;
+ stage.entityTail = e;
+
+ sscanf(line, "%*s %f %f %f %f", &e->sx, &e->sy, &e->ex, &e->ey);
+
+ e->health = 100;
+
+ e->x = e->sx;
+ e->y = e->sy;
+
+ e->tick = tick;
+ e->touch = touch;
+ e->flags = EF_PUSH;
+
+ texture[0] = loadTexture("gfx/alienF1.png");
+ texture[1] = loadTexture("gfx/alienB1.png");
+ e->texture = texture[0];
+
+
+ SDL_QueryTexture(e->texture, NULL, NULL, &e->w, &e->h);
+}
+
+static void tick(void)
+{
+ if (abs(self->x - self->sx) < PLATFORM_SPEED)
+ {
+   calcSlope(self->ex, self->ey, self->x, self->y, &self->dx, &self->dy);
+
+   self->dx *= PLATFORM_SPEED;
+   self->dy *= PLATFORM_SPEED;
+   self->texture = texture[0];
+ }
+
+ if (abs(self->x - self->ex) < PLATFORM_SPEED)
+ {
+   calcSlope(self->sx, self->sy, self->x, self->y, &self->dx, &self->dy);
+
+   self->dx *= PLATFORM_SPEED;
+   self->dy *= PLATFORM_SPEED;
+   self->texture = texture[1];
+ }
+}
```

```
+
+static void touch(Enemy* other)
+{
+ if (other == player)
+ {
+   other->health -= 1;
+
+ }
+}
\ No newline at end of file
diff --git a/src/entities.c b/src/entities.c
index b5827d9..20cb67c 100644
--- a/src/entities.c
+++ b/src/entities.c
@@ -29,10 +29,34 @@ static void addEntFromLine(char *line);

 void initEntities(void)
 {
- loadEnts("data/ents01.dat");
+
+ if (level == 1)
+ {
+   loadEnts("data/ents01.dat");
+ }
+ else if (level == 2)
+ {
+   loadEnts("data/ents02.dat");
+ }
+ else if (level == 3)
+ {
+   loadEnts("data/ents03.dat");
+ }
+ else if (level == 4)
+ {
+   loadEnts("data/ents04.dat");
+ }
+ else if (level == 5)
+ {
+   loadEnts("data/ents05.dat");
+ }
+ else
+ {
+   loadEnts("data/ents01.dat");
+ }
 }
```

```

-void doEntities(void)
+void doEntities()
 {
   Entity *e, *prev;

@@ -47,7 +71,7 @@ void doEntities(void)
        e->tick();
      }

-     move(e);
+                   move(e);

      if (e->health <= 0)
      {
@@ -60,7 +84,6 @@ void doEntities(void)
        free(e);
        e = prev;
      }
-
      prev = e;
    }

@@ -288,9 +311,21 @@ static void addEntFromLine(char *line)
   {
      initPlatform(line);
   }
- else if (strcmp(name, "PIZZA") == 0)
+ else if (strcmp(name, "ORB") == 0)
+ {
+    initOrb(line);
+ }
+ else if (strcmp(name, "ALIEN1") == 0)
+ {
+    initEnemy(line);
+ }
+ else if (strcmp(name, "ROCK1") == 0)
+ {
+    initRock1(line);
+ }
+ else if (strcmp(name, "ROCK2") == 0)
   {
-    initPizza(line);
+    initRock2(line);
   }
```

```
  else
  {
diff --git a/src/init.c b/src/init.c
index d341194..518e3ad 100644
--- a/src/init.c
+++ b/src/init.c
@@ -42,7 +42,7 @@ void initSDL(void)

     Mix_AllocateChannels(MAX_SND_CHANNELS);

- app.window = SDL_CreateWindow("Pete's Pizza Party 6", SDL_WINDOWPOS_UNDEFINED, SDL
+ app.window = SDL_CreateWindow("Astronuts Adventuere", SDL_WINDOWPOS_UNDEFINED, SDL

  SDL_SetHint(SDL_HINT_RENDER_SCALE_QUALITY, "linear");

@@ -55,10 +55,15 @@ void initSDL(void)

 void initGame(void)
 {
+ SDL_Texture* newGame = loadTexture("gfx/bg.jpg");
+ SDL_RenderCopy(app.renderer, newGame, NULL, NULL);
+ SDL_RenderPresent(app.renderer);
  initFonts();

  initSounds();

+ initMenu();
+
  loadMusic("music/one_0.mp3");

  playMusic(1);
diff --git a/src/map.c b/src/map.c
index 12c9291..c2557b9 100644
--- a/src/map.c
+++ b/src/map.c
@@ -32,7 +32,30 @@ void initMap(void)

  loadTiles();

- loadMap("data/map01.dat");
+ if (level == 1)
+ {
+    loadMap("data/map01.dat");
+ }
+ else if (level == 2)
+ {
+   loadMap("data/map02.dat");
+ }
+ else if (level == 3)
+ {
+   loadMap("data/map03.dat");
+ }
+ else if (level == 4)
+ {
+   loadMap("data/map04.dat");
+ }
+ else if (level == 5)
+ {
+   loadMap("data/map05.dat");
+ }
+ else
+ {
+   loadMap("data/map01.dat");
+ }
 }

 void drawMap(void)
@@ -78,7 +101,7 @@ static void loadTiles(void)

   for (i = 1 ; i <= MAX_TILES ; i++)
   {
-    sprintf(filename, "gfx/tile%d.png", i);
+    sprintf(filename, "gfx/tile_%d.png", i);

     tiles[i] = loadTexture(filename);
   }
diff --git a/src/menu.c b/src/menu.c
new file mode 100644
index 0000000..2192627
--- /dev/null
+++ b/src/menu.c
@@ -0,0 +1,56 @@
+
+#include "common.h"
+
+void initMenu(void)
+{
+ SDL_ShowCursor(1);
+ long then;
+ then = SDL_GetTicks();
+ //SDL_SetRenderDrawColor(app.renderer, 200, 53, 53, 235);
+
+ SDL_Texture* menuImg = IMG_LoadTexture(app.renderer, "gfx/menu.png");
+
+ int exit = 0;
+
+ while (!exit)
+ {
+   SDL_RenderCopy(app.renderer, menuImg, NULL, NULL);
+   SDL_RenderPresent(app.renderer);
+
+   doInput();
+
+   if (app.keyboard[SDL_SCANCODE_1])
+   {
+     level = 1;
+     exit = 1;
+   }
+   if (app.keyboard[SDL_SCANCODE_2])
+   {
+     level = 2;
+     exit = 1;
+   }
+   if (app.keyboard[SDL_SCANCODE_3])
+   {
+     level = 3;
+     exit = 1;
+   }
+   if (app.keyboard[SDL_SCANCODE_4])
+   {
+     level = 4;
+     exit = 1;
+   }
+   if (app.keyboard[SDL_SCANCODE_5])
+   {
+     level = 5;
+     exit = 1;
+
+   }
+   if (app.keyboard[SDL_SCANCODE_Q])
+   {
+     cleanup();
+   }
+   presentScene();
+ }
+ initStage();
+
+}
diff --git a/src/orb.c b/src/orb.c
new file mode 100644
index 0000000..2434752
--- /dev/null
+++ b/src/orb.c
@@ -0,0 +1,74 @@
+/*
+Copyright (C) 2015-2018 Parallel Realities
+
+This program is free software; you can redistribute it and/or
+modify it under the terms of the GNU General Public License
+as published by the Free Software Foundation; either version 2
+of the License, or (at your option) any later version.
+
+This program is distributed in the hope that it will be useful,
+but WITHOUT ANY WARRANTY; without even the implied warranty of
+MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
+
+See the GNU General Public License for more details.
+
+You should have received a copy of the GNU General Public License
+along with this program; if not, write to the Free Software
+Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
+
+*/
+
+#include "common.h"
+
+static void tick(void);
+static void touch(Entity *other);
+
+void initOrb(char *line)
+{
+ Entity *e;
+
+ e = malloc(sizeof(Entity));
+ memset(e, 0, sizeof(Entity));
+ stage.entityTail->next = e;
+ stage.entityTail = e;
+
+ sscanf(line, "%*s %f %f", &e->x, &e->y);
+
+ e->health = 1;
+
+ e->texture = loadTexture("gfx/spr.png");
+ SDL_QueryTexture(e->texture, NULL, NULL, &e->w, &e->h);
+ e->flags = EF_WEIGHTLESS;
+ e->tick = tick;
+ e->touch = touch;
+
+ stage.orbTotal++;
+}
+
+static void tick(void)
+{
+ self->value += 0.1;
+
+ self->y += sin(self->value);
+}
+
+static void touch(Entity *other)
+{
+ if (self->health > 0 && other == player )
+ {
+   self->health = 0;
+
+   stage.orbFound++;
+   stage.bulletCount += 5;
+
+   if (stage.orbFound == stage.orbTotal)
+   {
+     //levelCompleted(level);
+     playSound(SND_ORB_DONE, CH_ORB);
+   }
+   else
+   {
+     playSound(SND_ORB, CH_ORB);
+   }
+ }
+}
 diff --git a/src/platform.c b/src/platform.c
 index f9a0315..a83d239 100644
--- a/src/platform.c
+++ b/src/platform.c
@@ -21,6 +21,7 @@ Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA
02111-1307, USA.
 #include "common.h"

 static void tick(void);
+static void touch(Entity* other);

 void initPlatform(char *line)
 {
@@ -39,10 +40,12 @@ void initPlatform(char *line)
  e->y = e->sy;

  e->tick = tick;
+ e->touch = touch;

  e->texture = loadTexture("gfx/platform.png");
+
  SDL_QueryTexture(e->texture, NULL, NULL, &e->w, &e->h);
- e->flags = EF_SOLID+EF_WEIGHTLESS+EF_PUSH;
+ e->flags = EF_SOLID+EF_WEIGHTLESS;
 }

 static void tick(void)
@@ -63,3 +66,11 @@ static void tick(void)
     self->dy *= PLATFORM_SPEED;
  }
 }
+
+static void touch(Entity* other)
+{
+ if (other->flags == EF_WEIGHTLESS)
+ {
+    other->health -= 1;
+ }
+}
diff --git a/src/player.c b/src/player.c
index 489b69f..326e9ef 100644
--- a/src/player.c
+++ b/src/player.c
@@ -20,7 +20,7 @@ Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA
02111-1307, USA.

 #include "common.h"

-static SDL_Texture *pete[2];
+static SDL_Texture *astronut[7];

 void initPlayer(void)
 {
@@ -29,12 +29,23 @@ void initPlayer(void)
  stage.entityTail->next = player;
  stage.entityTail = player;

- player->health = 1;
+ player->health = 10;
+ stage.bulletCount = 100000;

- pete[0] = loadTexture("gfx/pete01.png");
- pete[1] = loadTexture("gfx/pete02.png");
+ astronut[0] = loadTexture("gfx/astroF1.png"); // stand faceing right
+ astronut[1] = loadTexture("gfx/astroB1.png"); // stand faceing left

- player->texture = pete[0];
+ astronut[2] = loadTexture("gfx/astroF2.png"); // walking faceing right
+ astronut[3] = loadTexture("gfx/astroB2.png"); // walking faceing left
+
+ astronut[4] = loadTexture("gfx/astroFJ.png"); // jump faceing right
+ astronut[5] = loadTexture("gfx/astroBJ.png"); // jump faceing left
+
+ astronut[6] = loadTexture("gfx/astroFS.png"); // shoot faceing right
+ astronut[7] = loadTexture("gfx/astroBS.png"); // shoot faceing left
+
+
+ player->texture = astronut[2];

  SDL_QueryTexture(player->texture, NULL, NULL, &player->w, &player->h);
 }
@@ -47,29 +58,55 @@ void doPlayer(void)
  {
    player->dx = -PLAYER_MOVE_SPEED;

-   player->texture = pete[1];
+   player->texture = astronut[3];
+   player->facing = LEFT;
  }

  if (app.keyboard[SDL_SCANCODE_D])
  {
    player->dx = PLAYER_MOVE_SPEED;

-   player->texture = pete[0];
+   player->texture = astronut[2];
+   player->facing = RIGHT;
```

```
   }

- if (app.keyboard[SDL_SCANCODE_I] && player->isOnGround)
+ if (app.keyboard[SDL_SCANCODE_SPACE] ) // jumping && player->isOnGround
   {
-    player->riding = NULL;
+    player->dy = -15;
+ }

-    player->dy = -20;

-    playSound(SND_JUMP, CH_PLAYER);
+ if (app.keyboard[SDL_SCANCODE_LSHIFT])
+ {
+    initBullet();
   }

- if (app.keyboard[SDL_SCANCODE_SPACE])
+ if (app.keyboard[SDL_SCANCODE_R] && player->health <= 0)
   {
-    player->x = player->y = 0;
+    initPlayer();
+ }
+

-    app.keyboard[SDL_SCANCODE_SPACE] = 0;
+ if (app.keyboard[SDL_SCANCODE_ESCAPE])
+ {
+    player->health = 0;
+    initMenu();
   }
-}
+
+ if (player->facing == RIGHT && player->dx == 0)
+ {
+    player->texture = astronut[0];
+ }
+ if (player->facing == LEFT && player->dx == 0)
+ {
+    player->texture = astronut[1];
+ }
+ if (player->facing == RIGHT && !player->isOnGround)
+ {
+    player->texture = astronut[4];
+ }
+ if (player->facing == LEFT && !player->isOnGround)
+ {
+    player->texture = astronut[5];
+ }
+}
\ No newline at end of file
diff --git a/src/sound.c b/src/sound.c
index 77e0cad..50ab74e 100644
--- a/src/sound.c
+++ b/src/sound.c
@@ -65,6 +65,6 @@ static void loadSounds(void)
    else {
      printf("Loaded sound successfully\n");
    }
- sounds[SND_PIZZA] = Mix_LoadWAV("sound/90134__pierrecartoons1979__found-item.ogg")
- sounds[SND_PIZZA_DONE] = Mix_LoadWAV("sound/449069__ricniclas__fanfare.ogg");
+ sounds[SND_ORB] = Mix_LoadWAV("sound/90134__pierrecartoons1979__found-item.ogg");
+ sounds[SND_ORB_DONE] = Mix_LoadWAV("sound/449069__ricniclas__fanfare.ogg");
 }
diff --git a/src/stage.c b/src/stage.c
index dc4461c..bf0345b 100644
--- a/src/stage.c
+++ b/src/stage.c
@@ -51,9 +51,8 @@ static void logic(void)

 static void draw(void)
 {
- SDL_SetRenderDrawColor(app.renderer, 128, 0, 255, 255);
- SDL_RenderFillRect(app.renderer, NULL);
-
+ SDL_Texture* newGame = loadTexture("gfx/bg.jpg");
+ SDL_RenderCopy(app.renderer, newGame, NULL, NULL);
  drawMap();

  drawEntities();
@@ -75,5 +74,9 @@ static void drawHud(void)
  SDL_RenderFillRect(app.renderer, &r);
  SDL_SetRenderDrawBlendMode(app.renderer, SDL_BLENDMODE_NONE);

- drawText(SCREEN_WIDTH - 5, 5, 255, 255, 255, TEXT_RIGHT, "PIZZA %d/%d", stage.pizza
+ drawText(SCREEN_WIDTH - 5, 5, 255, 255, 255, TEXT_RIGHT, "ORBS %d/%d", stage.orbFo
+ drawText(SCREEN_WIDTH - 500, 5, 255, 255, 255, TEXT_RIGHT, "HEALTH %d", player->he
+ drawText(SCREEN_WIDTH - 1000, 5, 255, 255, 255, TEXT_RIGHT, "X %.1f Y %.1f", playe
+ drawText(SCREEN_WIDTH - 100, 5, 255, 255, 255, TEXT_RIGHT, "%d Bullet LEFT", stage
+ //drawText(SCREEN_WIDTH - 1000, 5, 255, 255, 255, TEXT_RIGHT, "LEVEL %d", level);
```

```
 }
diff --git a/src/structs.h b/src/structs.h
index 0ad62ab..56651a6 100644
--- a/src/structs.h
+++ b/src/structs.h
@@ -19,7 +19,8 @@ Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
  */
 
 typedef struct Texture Texture;
-typedef struct Entity Entity;
+typedef struct Entity Entity;
+typedef struct Entity Enemy;
 
 typedef struct {
   void (*logic)(void);
@@ -53,18 +54,21 @@ struct Entity {
   float dy;
   int health;
   int isOnGround;
+  int count;
+  int facing;
   float value;
-  SDL_Texture *texture;
+  SDL_Texture* texture;
   void (*tick)(void);
-  void (*touch)(Entity *other);
+  void (*touch)(Entity* other);
   long flags;
-  Entity *riding;
-  Entity *next;
+  Entity* riding;
+  Entity* next;
 };
 
 typedef struct {
   SDL_Point camera;
   int map[MAP_WIDTH][MAP_HEIGHT];
   Entity entityHead, *entityTail;
-  int pizzaTotal, pizzaFound;
-} Stage;
+  int orbTotal, orbFound;
+  int bulletCount;
+} Stage;
\ No newline at end of file
```

```
diff --git a/src/textures.c b/src/textures.c
index 33b1e99..36f10cb 100644
--- a/src/textures.c
+++ b/src/textures.c
@@ -49,7 +49,7 @@ static void addTextureToCache(char *name, SDL_Texture *sdlTexture)
   texture->texture = sdlTexture;
 }

-SDL_Texture* loadTexture(char* filename)
+SDL_Texture* loadTexture(char* filename[6])
 {
   SDL_Texture *texture;
```

Sources and References