

C++ (DX ライブラリ) を用いて作成

1. 1 動機

将来の夢がゲームプログラマーであり、そのための練習がしたかった。また、単純にゲームを作ってみたいと思っていたので、課題研究のテーマにゲーム開発を選びました。

1. 2 内容

今回作ったゲームは、主人公を操作してマップ内の”●”を制限時間 (60 秒) 以内にいくつ回収できるかを競うミニゲームです。”●”は全部で 20 個配置されています。

マップのシステムには『マップチップ』というシステムを採用しています。これは、マップをドットのように細かく分割して、その一つ一つに役割 (通れる、通れない等) を持たせ、それらを組み合わせる一つのマップを形成するシステムです。

例えば図 1 の場合、上側を壁とする場合には (仮に関数を `int mapchip[25]` としたとき) `mapchip[0]~mapchip[4]` までに 1 を与えます。事前に 1 を与えられたマップチップは通さないと思うようプログラムで指示を出しているため、もうこれで主人公キャラは 0~4 のチップの中には入ることが出来なくなります。こうすることで壁ができます。

図 1

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

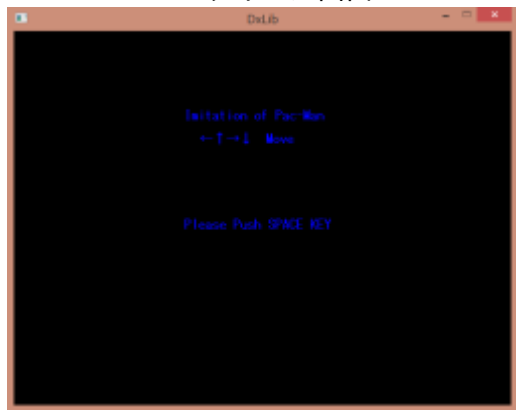
右の画像が今回作成したゲームのマップチップのデータです。0 が通行可、2 が通行不可です。マップは 22x22 だったのでチップの数は 484 になっています。

マップ自体は別のソフト (Illustrator) を使用して、正方形の画像として作成しました。そのマップを縦 22、横 22 で分割して、それとにらめっこしながら、ここはこうしよう、こっちはこうしよう、と考えながら、484 マスを一つ一つ丁寧に作りました。

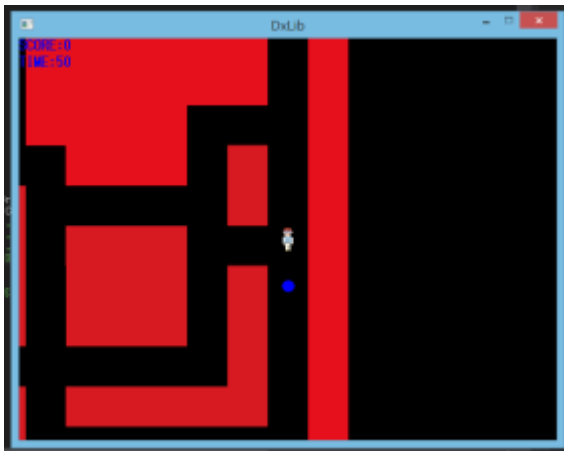
```
int mapchip[484] = {
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2,
2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2,
2, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2,
2, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2,
2, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 2,
2, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 2, 2, 2, 0, 0, 2,
2, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 0, 0, 0, 0, 2, 2, 2, 2, 2, 0, 0, 2,
2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
2, 0, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2,
2, 0, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2,
2, 0, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 2,
2, 0, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 2,
2, 0, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 2,
2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
};
```

1. 3 結果

▽タイトル画面



▽ゲーム画面



C++ (DX ライブラリ) を用いて作成

2. 1 動機

進学先でゲームプログラムを習う前に独自で調べゲームを作り、さらにプログラムへの興味関心を高めるため課題研究でゲームを作ることにしました。

2. 2 内容

作った作品は1 マップ探索型のホラーゲームです。光を集め、脱出をするというものです。こだわった部分は他のゲームにあまりない歩数制限です。主人公が移動をするごとに歩数カウンタが減っていきゲームオーバーになります。この歩数カウンタの減少は主人公のx,yの増減に伴い減少するようにしました。

```
if (mapchip[i]!=0){ //mapchip[i]が0ではないとき
  x+=2; //xに2足す
  cnt--; //カウントを1減らす
}
if (mapchip[i] != 0){ //mapchip[i]が0ではないとき
  x-=2; //xから2引く
  cnt--; //カウントを1減らす
}
if (mapchip[i] != 0){ //mapchip[i]が0ではないとき
  y+=2; //yに2足す
  cnt--; //カウントを1減らす
}
if (mapchip[i]!=0){ //mapchip[i]が0ではないとき
  y-=2; //yから2引く
  cnt--; //カウントを1減らす
}
```

そしてもう1つこだわった部分が奥行きです。プログラムでは表示が古い順に奥に表示されます。1つの画像で前の壁と奥の壁があるマップ(画像2)を使用しました。

そのため前の壁より主人公のほうが出てしまう不具合がありました。ここでPhotoshopを使い前の壁を切り取り別の画像に変え、障子を1つ切り取った画像を作りました。そして、プログラムによって座標などの調整を行い、奥行きの表示に成功しました。1つ切り取った障子は出口が開く演出のために使用しました。これらのことは実際にプレイ画面を移す時に注目して見てください。



2. 3 結果



ゲーム画面
タイトル画面



Java (Eclipse) を用いて作成

3. 1 動機

Minecraft というゲームの追加要素を作成するのに java というプログラミング言語を使用するからということと、自分でアクションゲームを作り仕組みを理解したかったからです。

3. 2 内容

作成したゲームは固定画面のアクションゲームでプレイヤーを操作し、飛んでくる障害物のブロックを避けてスコアを稼ぐというものです。

工夫した点は

- ・加速度や摩擦係数、重力などを取り入れることによってプレイヤーの移動が少しずつ速くなり止まるときに少し滑り、ジャンプが滑らかになるようにした。
- ・乱数とスコアの計算を用いて障害物の生成をランダムにし、スコアが上がるにつれて難易度が上がるようにした。

ことです。

苦労した点は

- ・すべてのオブジェクトを考えながら座標の管理をしなければならないこと。
- ・プレイヤーが画面端に行くとそこで停止するようにすること
- ・あたり判定をプレイヤーが快適にプレイできるように考え、プログラムを最適化する

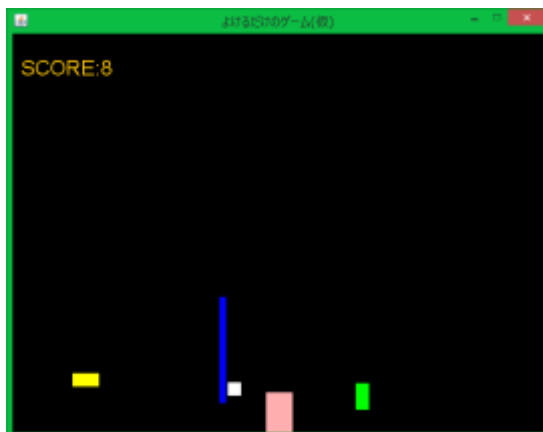
ことです。

3. 3 結果

タイトル画面→



←ゲーム画面



ゲームオーバー画面→



4 まとめ

java (Eclipse 使用) と C++ (DX ライブラリ使用) の違い

この二つはおおよそ同じような言語であり、出来ることの幅も近い。そもそも Java は C 言語の構文を主に引き継いだプログラミング言語であり、C++ は C 言語の発展形である。

- ・用途の違い

Java は汎用的で OS に依存しないため、様々な環境で使える。

C++ はより高度なパフォーマンスを要求されるシステムを作ることに適している。

- ・ライブラリの違い

Java は import、C++ では #include という違いがある。

5 感想

今回の課題研究で残念だったことは、始めた当初作ろうとしていたゲームを断念してしまったことです。僕は最初、RPG ゲームを作ろうとしていました。しかし、キャラを動かすというゲームの基本となるシステムを作るだけで、自分が思っていた以上の時間がかかってしまいました。簡単なことでも、初めての試みであればその難易度は大きくなります。RPG を作るために必要な最低限のシステムと、残りの時間を見比べたとき、僕はどう頑張っても時間が足りなくなると気付き、技量不足と皮算用を痛感しました。それでも時間は待ってくれません。悔しさを胸に、僕はすでに完成していたシステムを流用できるようなゲーム内容の変更を決断しました。

僕は将来、ゲームプログラマーになりたいと思っています。そのために必要なものをこの課題研究を通して思い知らされました。それは経験と諦めです。作るまでにかかなりの時間を要したシステムの数々も、仕組み自体は簡単なもので、一度やり方がわかれば作ることができるようなものばかりです。それを見つけるための経験が、僕には圧倒的に不足していました。

また、諦めも肝心です。仕事には、ゲーム作りには期限があります。最高のものを作れるように努力することは、そもそも前提条件です。それを踏まえたうえで、それでも決められた期間内に完成させることが大事なのです。より時間をかければ、より良いものができるのは当たり前です。だからこそ、決められた時間内に収め、納めることが大切なのだと実感しました。

課題研究が始まりたての時は自分で作れるのかと心配でした。でも実際に調べながら進めていくと理解できることが多く、進めやすかったです。わからないところがあっても他のメンバーに聞くことで解決できることが多々ありました。この課題研究で学べたことは、プログラミングのアルゴリズムです。どこにどう入れればしっかり動くのか検定の本やネットで調べつつ1から組むことで理解していくことができました。1からのプログラムは検定の勉強ではできなかったのでもいい経験になったと思います。この経験は進学先、就職先で活かしていきたいと思います。

今回、Java というプログラミング言語を一から勉強しゲームを作り、様々なことが学べました。ネットや本などの資料を見る、講演会などに参加するなどして Java のことを勉強することができ、ある程度使えるようになりました。ゲームを作り、テストと改良を繰り返していくことの苦勞を知りました。様々な人にプレイしてもらいどの部分がわかりにくいのか、どうしたらプレイしたくなるかなどを行ってもらったのでより良い方向に修正できました。このことから、ゲームを作る人とプレイする人の気持ちがよく理解することができました。

