

4. ゲームを作ろう

ゲームの作り方

ゲームをはじめとしたソフトウェアを開発するには、まず『設計』という作業をします。いきなりプログラムを書き始めるのではなく、どのようなソフトウェアを作りたいのか、最初に考えることはとても重要です。

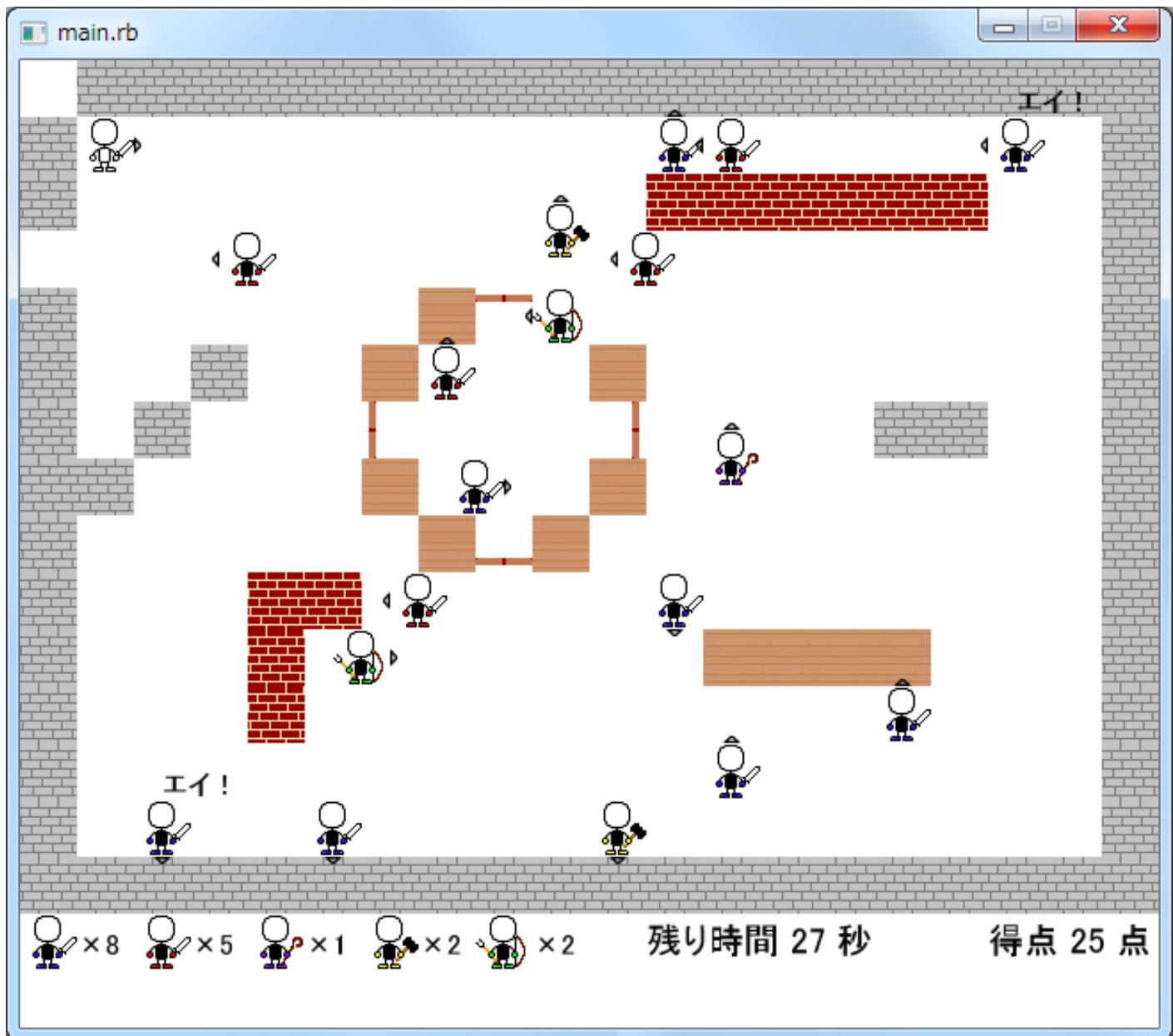
- 作りたいゲームをイメージしてみよう
- イメージしたものを紙に描いてみよう
- キャラクターをデザインしよう
- 背景をデザインしよう
- キャラクターの動きを決めよう
- キーボードやマウスを使ったキャラクターの動かし方を決めよう
- プログラムの構成を考えよう

今回のゲーム

主人公を操作して敵を倒すゲームです。

敵をすべて倒すとクリアです。

ブロックや、ドアなどの障害物があります。



- 登場人物
- 主人公
- 普通の敵
- 追いかけてくる敵
- 障害物を作る敵
- 障害物を壊す敵

-  遠距離攻撃をする敵

プログラムの作成

設計ができたら、実際にプログラムを作成していきます。

ソフトウェアはプログラムが書かれたいくつものファイルによって構成されています。

1. エクスプローラで確認してみよう。
2. エディタで『main.rb』を開いてみよう。プログラムが書かれていることがわかります。
3. プログラムは **半角英数字と記号** で書きます。全角と半角に気をつけよう。また、大文字、小文字も区別するので注意しよう。
4. 何かプログラムを書いたら、必ず保存を実行しよう。また、ファイル名には半角英数字を使うことに気をつけよう。

ソフトウェアの実行

作成したソフトウェアを実行するには、コマンドプロンプトを使います。エクスプローラでダブルクリックするのではありません。

コマンドプロンプトを開いて、以下の文字列（コマンドと呼びます）を入力しエンターキーを押すと、プログラムの実行がはじまります。

```
ruby main.rb
```

最初は何も反応がありませんが、プログラムを徐々に追加していくとゲームとして動くようになります。

ゲームの作成

今まで学んだ知識を使いながら、一緒にゲームを作っていきましょう。

ステップ1. ウィンドウとプレーヤーを表示する

最初は、ゲームのベースとなるウィンドウと主人公を表示してみましょう。

main.rb

『main.rb』をエディタで開いて、以下のようにプログラムを追加してみてください。プログラムを入力したら、忘れずに保存しよう。

```
1 require_relative "./bomber"
2
3 Window.height += 64
4
5 player = Bomber::Player.new(1, 1, 0)
6
7 player.on(:start) do
8   end
```

bomber/player.rb

エディタで新規作成を選択して、新しいファイルを作ります。

以下のようにプログラムを入力し、bomberディレクトリに『player.rb』として保存します。

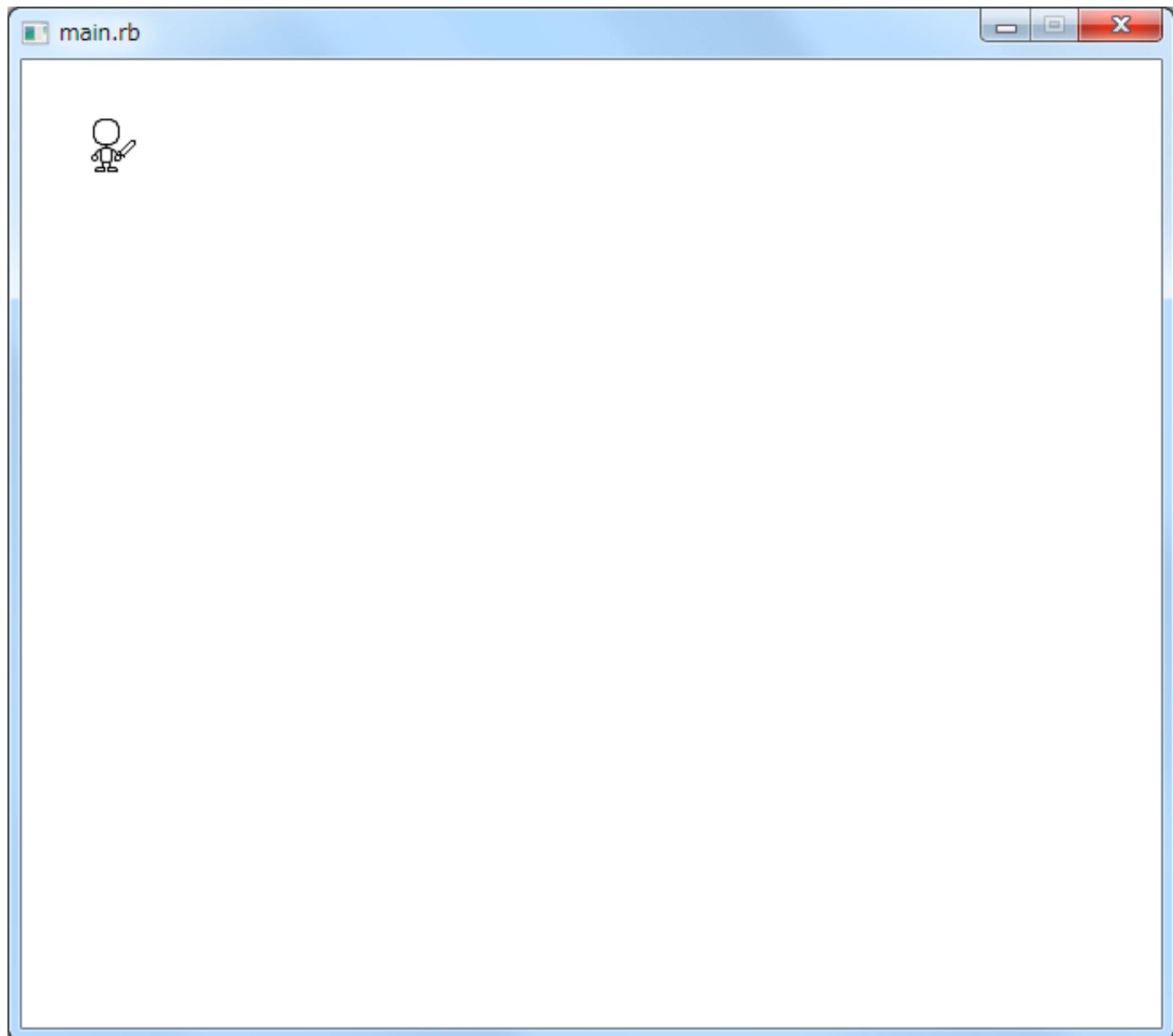
```
1 module Bomber
2   class Player < Bomber::Character
3     attr_accessor :guide, :agl
4     def initialize(x, y, angle)
5       super(costume_lists, x, y, angle)
6       @agl = :right
7     end
8
9     def costume_lists
10      ["./image/hito.png"]
11    end
12
13    def move(move_angle=:right)
14      if self.agl == move_angle
15        self.send("move_#{move_angle.to_s}.to_sym")
16        sleep 0.1
17      end
18      angle_shift(move_angle)
19    end
20
21    def lose
22      super
```

```
23 |     sleep 3
24 |     exit
25 |   end
26 | end
27 | end
```

プログラムを保存したら、コマンドプロンプトを開いて、以下のコマンドを入力し、エンターキーを押してみましょう。

```
ruby main.rb
```

以下のようなウィンドウが開けば、ステップ1は終了です。



ウィンドウが開かない場合はプログラムが間違っているので見直しましょう。
コマンドプロンプトにエラーが出ているかもしれないで、確認してみましょう。

ステップ2.主人公を動かす

次は、主人公をキーボードを使って動かしてみましょう。
また、主人公が向いている方向を示すガイド機能も作りましょう。

bomber/guide.rb

まず、主人公が向いている方向を示すガイド機能を作ります。

エディタで新規作成を選択して、新しいファイルを作ります。

以下のようにプログラムを入力し、bomberディレクトリに『guide.rb』として保存します。

```
1 module Bomber
2   class Guide < Bomber::Character
3     def initialize(target)
4       @target = target
5       super(costume_lists, @target.x, @target.y, 0)
6     end
7
8     def costume_lists
9       ["../image/up.png",
10      "../image/down.png",
11      "../image/left.png",
12      "../image/right.png"]
13   end
14
15   def trace
16     agl = @target.agl
17     self.x, self.y = @target.x, @target.y
18     case agl
19     when :up
20       self.y -= 4
21       self.image = @costumes[0]
22     when :down
23       self.y += 32
24       self.image = @costumes[1]
25     when :left
26       self.x -= 4
27       self.image = @costumes[2]
28     when :right
29       self.x += 32
30       self.image = @costumes[3]
31     end
32   end
33 end
34 end
```

ここで、

```
def costume_lists
  ["../image/up.png",
  "../image/down.png",
```

```
"../image/left.png",
"../image/right.png"]
end
```

は、ガイドに使用する画像を指定する処理です。

```
case agl
when :up
  self.y -= 4
  self.image = @costumes[0]
```

は、主人公の位置からみて、どの位置にガイドを表示するかを指定する処理です。xやyは座標です。

bomber/player.rb

上記で作成したガイドを主人公のまわりに表示されるようにします。

キーボードが押されたら、押された方向に合わせてガイドが表示されるようにします。
以下となるように、『player.rb』にプログラムを追加します。

```
4  def initialize(x, y, angle)
5    super(costume_lists, x, y, angle)
6    @agl = :right
7    @agl = :right
8    @guide = Bomber::Guide.new(self)
9    @guide.trace
10   end
```

main.rb

最後に、キーボードが押されたときにキャラクターを動かす処理を追加します。

『main.rb』をエディタで開いて、以下となるようにプログラムを変更してください。

```
1 require_relative "./bomber"
2
3 Window.height += 64
4
5 $score = 0
6
7 $all_obj = Array.new
8 $hit_obj = Array.new
9
10 player = Bomber::Player.new(1, 1, 0)
11 $all_obj << player
12 $all_obj.flatten!
13
14 $hit_obj << player
15 $hit_obj.flatten!
```

```
16
17 player.on(:start) do
18   on(:key_down, K_RIGHT) do
19     self.move(:right)
20   end
21
22   on(:key_down, K_LEFT) do
23     self.move(:left)
24   end
25
26   on(:key_down, K_UP) do
27     self.move(:up)
28   end
29
30   on(:key_down, K_DOWN) do
31     self.move(:down)
32   end
33
34 end
```

ここで、

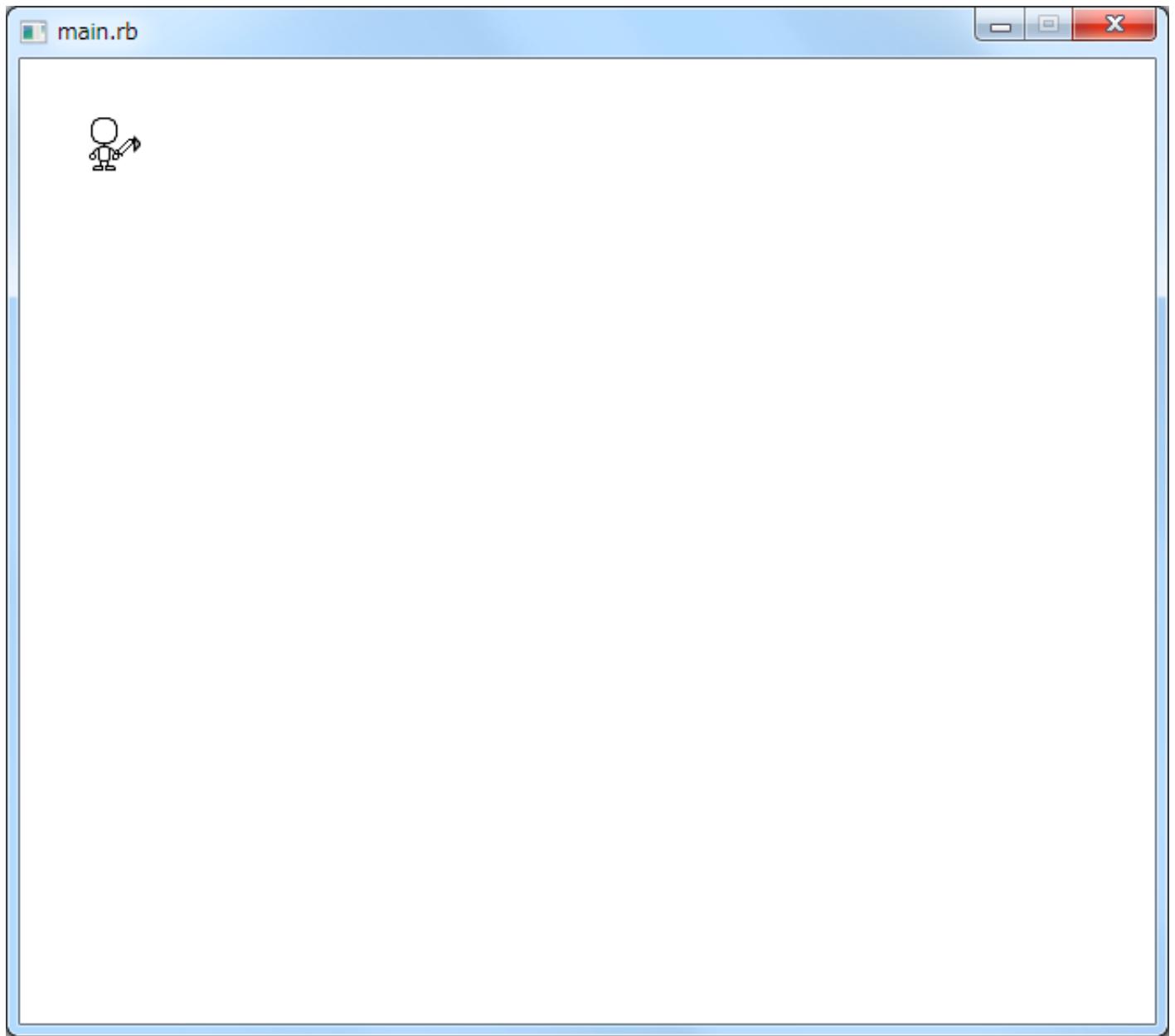
```
on(:key_down, K_RIGHT) do
  self.move(:right)
end
```

は、キーボードの「→」が押されたとき、主人公を右に動かすという処理です。

プログラムを保存したら、コマンドプロンプトを開いて、以下のコマンドを入力し、エンターキーを押してみましょう。

```
ruby main.rb
```

以下のようなウィンドウが開いて、キーボードで主人公が動くことを確認できたら、ステップ2は終了です。



ウィンドウが開かない、主人公が動かないなどは、プログラムが間違っているので見直しましょう。

コマンドプロンプトにエラーが出ているかもしれないで、確認してみましょう。

ステップ3. ウィンドウにブロックを配置する

次は、殺風景なウィンドウにブロックを配置してみましょう。

bomber/block.rb

まず、ブロックを表現するオブジェクトを作ります。

エディタで新規作成を選択して、新しいファイルを作ります。

以下のようにプログラムを入力し、bomberディレクトリに『block.rb』として保存します。

長いので、間違えないように注意して入力しよう。

```
1 module Bomber
2   class Block < Bomber::Character
3     attr_accessor :col
4     def initialize(data)
5       if data.class == Hash
6         @col = data['col']
7         super("../image/block_#{data['col'].png}", data["x"].to_i, data["y"].to_i, 0)
8       elsif data.class == Array
9         super(costume_lists, data[0], data[1], 0)
10      end
11      self.z = 0
12    end
13
14    def destroy
15      return if self.wall?
16      $hit_obj -= [self]
17      $all_obj -= [self]
18      fire = Bomber::Fire.new(*self.current_block)
19      self.vanish
20      sleep 0.1
21      fire.vanish
22    end
23
24    def wall?
25      self.current_y_block == 0 or self.current_y_block == HEIGHT or self.current_x_block ==
26    end
27
28    def costume_lists
29      ["../image/block_stone.png",
30       "../image/block_brick.png",
31       "../image/block_wood.png"]
32    end
33
34    def collar_lists
35      ["stone", "brick", "wood"]
36    end
37
38    def add_event
39      if Input.mouse_push?(M_LBUTTON)
40        $blocks -= [self]
41        self.vanish
42      elsif Input.mouse_push?(M_RBUTTON)
43        next_costume
44      end
45    end
46  end
47
```

```
44     puts self.image.inspect
45   else
46   end
47 end
48 end
49 end
```

bomber/door.rb

ブロックだけでなく、主人公が開けることができるドアオブジェクトも作りましょう。エディタで新規作成を選択して、新しいファイルを作ります。以下のようにプログラムを入力し、bomberディレクトリに『door.rb』として保存します。

```
1 module Bomber
2   class Door < Bomber::Character
3     attr_accessor :close
4     def initialize(*data)
5       @close = true
6       super(costume_lists(data[2]), data[0].to_i, data[1].to_i, 0)
7       self.z = 0
8     end
9
10    def costume_lists(data=:up)
11      ["../image/door_#{data.to_s}_close.png",
12       "../image/door_#{data.to_s}_open.png"]
13    end
14
15    def lose
16      next_costume
17      @close = !@close
18      if @close
19        $hit_obj << self
20      else
21        $hit_obj -= [self]
22      end
23    end
24
25    def destroy
26      return if self.current_y_block == 0 or self.current_y_block == HEIGHT or self.current_x_
27      $hit_obj -= [self]
28      $all_obj -= [self]
29      fire = Bomber::Fire.new(*self.current_block)
30      self.vanish
31      sleep 0.1
32      fire.vanish
33    end
34  end
35 end
```

main.rb

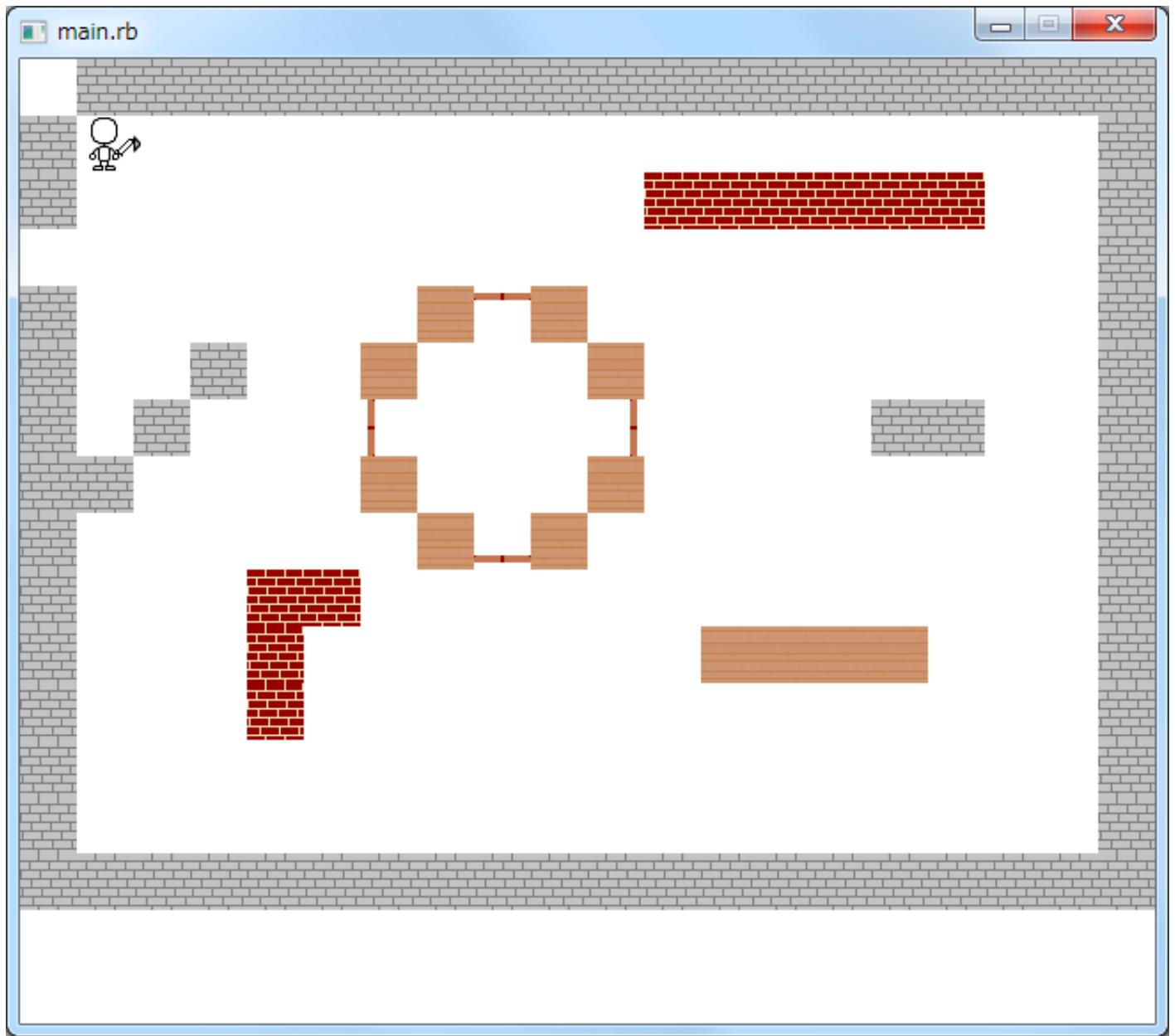
最後に、ウィンドウにブロックとドアを配置する処理を追加します。
『main.rb』をエディタで開いて、以下となるようにプログラムを変更してください。

```
1 require_relative "./bomber"
2
3 Window.height += 64
4
5 $score = 0
6
7 $all_obj = Array.new
8 $hit_obj = Array.new
9
10 $blocks = Array.new
11 datas = YAML.load_file(File.expand_path("../config.yml", __FILE__))
12 datas.each do |data|
13   $blocks << Block.new(data)
14 end
15
16 doors = Array.new
17 doors << door1 = Door.new(8, 4, :up)
18 doors << door2 = Door.new(6, 6, :left)
19 doors << door3 = Door.new(10, 6, :right)
20 doors << door4 = Door.new(8, 8, :down)
21
22 player = Player.new(1, 1, 0)
23 $all_obj << player
24 $all_obj << $blocks
25 $all_obj << doors
26 $all_obj.flatten!
27
28 $hit_obj << $blocks
29 $hit_obj << doors
30 $hit_obj << player
31 $hit_obj.flatten!
```

プログラムを保存したら、コマンドプロンプトを開いて、以下のコマンドを入力し、エンターキーを押してみましょう。

```
ruby main.rb
```

以下のようなウィンドウが開けば、ステップ3は終了です。
念のため、主人公がキーボードで操作できるかも確認してみましょう。



ウィンドウが開かない場合はプログラムが間違っているので見直しましょう。
コマンドプロンプトにエラーがでているかもしれないの、確認してみましょう。

ステップ4.敵を作る

次は、主人公以外のキャラクターを作つてみましょう。

bomber/enemy_normal.rb

敵を表現するオブジェクトを作ります。

エディタで新規作成を選択して、新しいファイルを作ります。

以下のようにプログラムを入力し、bomberディレクトリに『enemy_normal.rb』として保存します。

```
1 module Bomber
2   class EnemyNormal < Bomber::Character
3     attr_accessor :guide, :agl
4     def initialize(x, y, angle, delay=0)
5       super(costume_lists, x, y, angle)
6       @score = 5
7       @agl = :right
8       @delay = delay
9       @guide = Bomber::Guide.new(self)
10      @guide.trace
11    end
12
13   def costume_lists
14     ["../image/ene.png"]
15   end
16
17   def action_list
18     [:up, :down, :right, :left]
19   end
20
21   def auto
22     super
23     random_move
24   end
25
26   def random_move
27     num = rand(4)
28     sleep @delay
29     sleep 0.3
30     self.send("move_#{action_list[num].to_s}".to_sym)
31     sleep 0.2
32     self.send("move_#{action_list[num].to_s}".to_sym)
33     reject_half
34   end
35 end
36 end
```

main.rb

ウィンドウに敵を配置する処理を追加します。

『main.rb』をエディタで開いて、以下となるようにプログラムを変更してください。

```

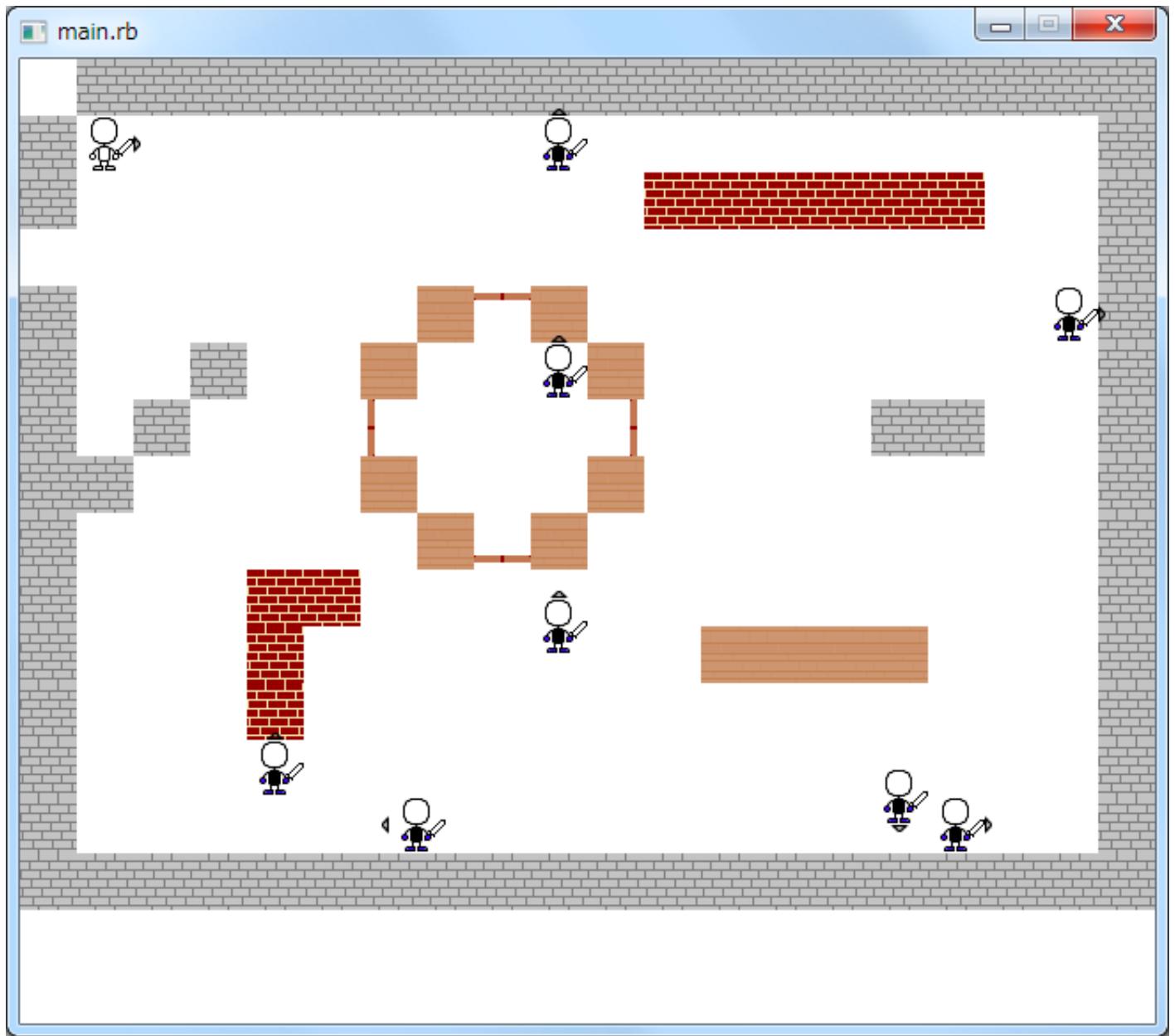
22 player = Bomber::Player.new(1, 1, 0)
23 $all_obj << player
24
25 $enemy = Array.new
26 $enemy << Bomber::EnemyNormal.new( 1, 13, 0, 0.3)
27 $enemy << Bomber::EnemyNormal.new( 5, 12, 0, 0.4)
28 $enemy << Bomber::EnemyNormal.new( 8, 7, 0, 0)
29 $enemy << Bomber::EnemyNormal.new(10, 8, 0, 0.2)
30 $enemy << Bomber::EnemyNormal.new(12, 1, 0, 0.1)
31 $enemy << Bomber::EnemyNormal.new(13, 13, 0, 0.1)
32 $enemy << Bomber::EnemyNormal.new(15, 11, 0, 0.2)
33 $enemy << Bomber::EnemyNormal.new(18, 1, 0, 0)
34
35 $enemy.each do |ene|
36   ene.on(:start) do
37     while self.active do
38       if self.active
39         self.auto
40       end
41     end
42   end
43 end
44
45 $all_obj << $blocks
46 $all_obj << $enemy
47 $all_obj << doors
48 $all_obj.flatten!
49
50 $hit_obj << $blocks
51 $hit_obj << $enemy
52 $hit_obj << doors
53 $hit_obj << player
54 $hit_obj.flatten!

```

プログラムを保存したら、コマンドプロンプトを開いて、以下のコマンドを入力し、エンターキーを押してみましょう。

`ruby main.rb`

以下のようなウィンドウが開いて、主人公のほかに敵が表示されていれば、ステップ4は終了です。



ウィンドウが開かない場合はプログラムが間違っているので見直しましょう。
コマンドプロンプトにエラーが出ているかもしれない、確認してみましょう。

ステップ5.攻撃する、攻撃される

bomber/fire.rb

主人公が攻撃したときに火を吹くようにします。

エディタで新規作成を選択して、新しいファイルを作ります。

以下のようにプログラムを入力し、bomberディレクトリに『fire.rb』として保存します。

```
1 module Bomber
2   class Fire < Bomber::Character
3     def initialize(x, y)
4       super("../image/fire.png", x, y, 0)
5     end
6   end
7 end
```

bomber/player.rb

主人公が攻撃する処理を追加します。

『player.rb』をエディタで開いて、以下となるようにプログラムを追加してください。

```
15 def attack
16   super
17   if all_enemy.empty?
18     sleep 3
19     exit
20   end
21 end
```

bomber/enemy_normal.rb

敵が攻撃する処理を追加します。

『enemy_normal.rb』をエディタで開いて、以下となるように33行目のプログラムを追加してください。

```
32   reject_half
33   attack if rand(5) == 0
34 end
```

main.rb

最後に、キーボードのスペースを押したら、主人公が攻撃する処理を追加します。

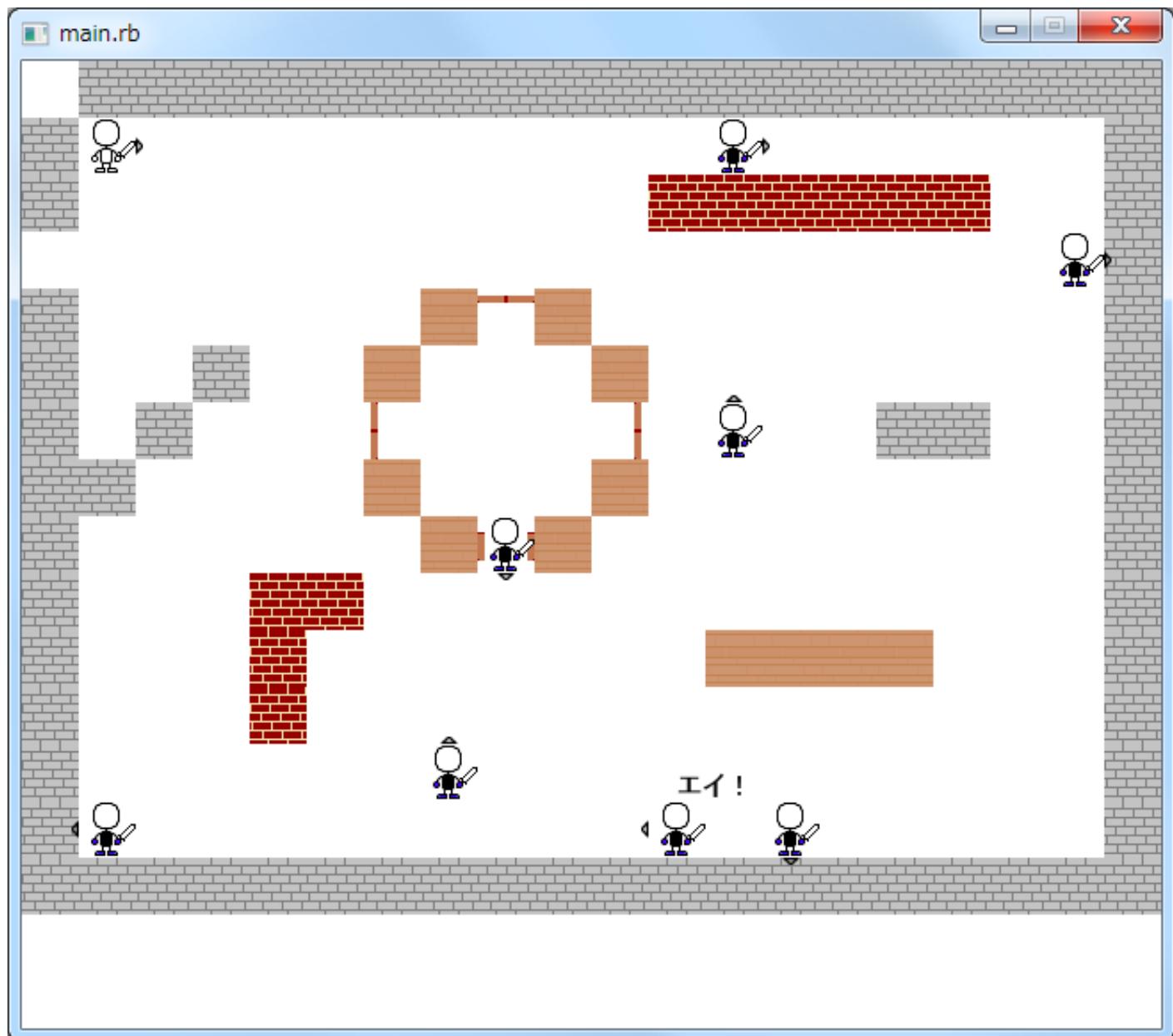
『main.rb』をエディタで開いて、以下となるようにプログラムを変更してください。

```
73 |   on(:key_push, K_SPACE) do
74 |     attack
75 |   end
76 | end
```

プログラムを保存したら、コマンドプロンプトを開いて、以下のコマンドを入力し、エンターキーを押してみましょう。

```
ruby main.rb
```

以下のようなウィンドウが開いて、スペースキーを押したら主人公が攻撃するようになれば、ステップ5は終了です。



ウィンドウが開かない場合はプログラムが間違っているので見直しましょう。
コマンドプロンプトにエラーが出ているかもしれないで、確認してみましょう。

ステップ6.ゲームクリアとゲームオーバー

bomber/game_clear.rb

ゲームクリアの画面を作ります。

エディタで新規作成を選択して、新しいファイルを作ります。

以下のようにプログラムを入力し、bomberディレクトリに『game_clear.rb』として保存します。

```
1 module Bomber
2   class GameClear < Bomber::Character
3     def initialize
4       super("../image/gameclear.png", 0, 0, 0)
5       self.z = 20
6     end
7   end
8 end
```

bomber/game_over.rb

ゲームオーバーの画面を作ります。

エディタで新規作成を選択して、新しいファイルを作ります。

以下のようにプログラムを入力し、bomberディレクトリに『game_over.rb』として保存します。

```
1 module Bomber
2   class GameOver < Bomber::Character
3     def initialize
4       super("../image/gameover.png", 0, 0, 0)
5       self.z = 20
6     end
7   end
8 end
```

bomber/player.rb

主人公が攻撃してすべての敵がいなくなったらゲームクリア、主人公が攻撃を受けたらゲームオーバーを表示するように処理を追加します。

『player.rb』をエディタで開いて、以下となるようにプログラムを変更してください。

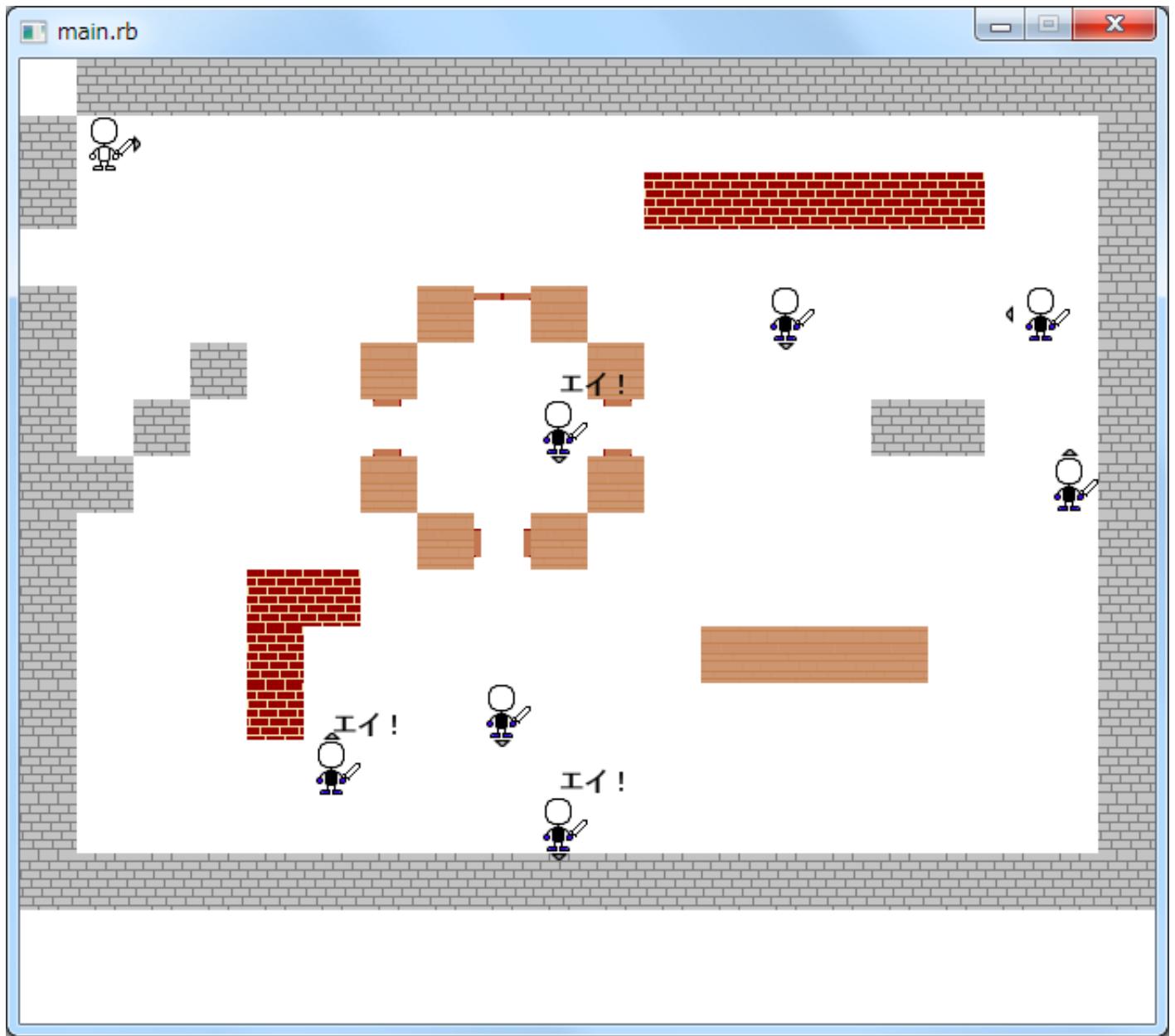
```
15 def attack
16   super
17   if all_enemy.empty?
18     sleep 1
19     gameclear = Bomber::GameClear.new
20     gameclear.z = 10
```

```
21     sleep 3
22     exit
23   end
24 end
25
26 def move(move_angle=:right)
27   if self.agl == move_angle
28     self.send("move_#{move_angle.to_s}").to_sym)
29     sleep 0.1
30   end
31   angle_shift(move_angle)
32 end
33
34 def lose
35   super
36   sleep 1
37   gameover = Bomber::GameOver.new
38   gameover.z = 10
39   sleep 3
40   exit
41 end
```

プログラムを保存したら、コマンドプロンプトを開いて、以下のコマンドを入力し、エンターキーを押してみましょう。

```
ruby main.rb
```

以下のようなウィンドウが開いて、ゲームクリア、ゲームオーバーが確認できたら、ステップ6は終了です。



ウィンドウが開かない場合はプログラムが間違っているので見直しましょう。
コマンドプロンプトにエラーがでているかもしれないの、確認してみましょう。

ステップ7.敵のバリエーションを増やす

bomber/enemy_hammer.rb

bomber/enemy_shooter.rb

bomber/enemy_trace.rb

bomber/enemy_wizard.rb

bomber/arrow.rb

main.rb

ウィンドウに新しい敵を配置する処理を追加します。

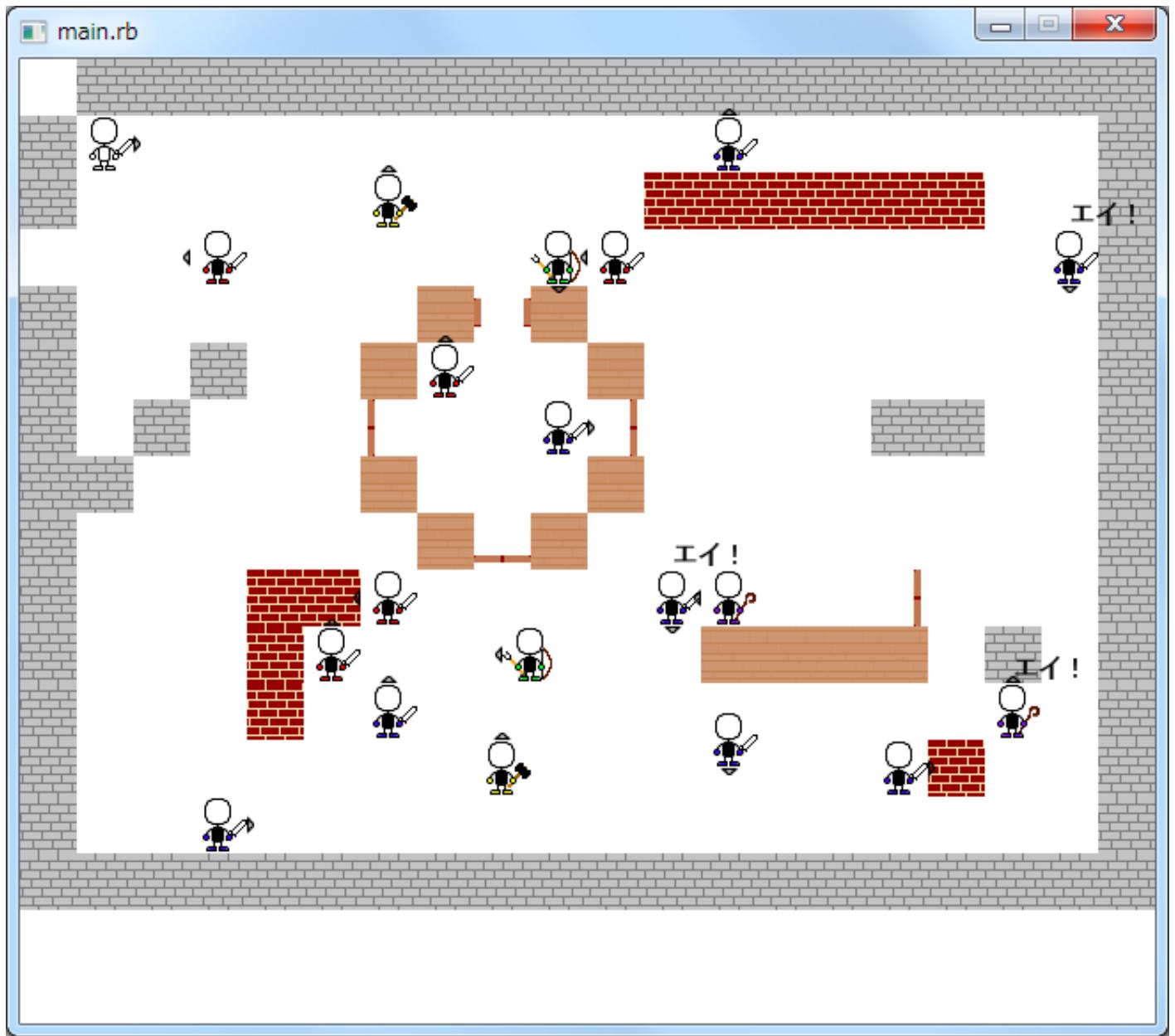
『main.rb』をエディタで開いて、以下となるようにプログラムを変更してください。

```
33 $enemy << Bomber::EnemyNormal.new(18, 1, 0, 0)
34 $enemy << Bomber::EnemyTrace.new( 5, 11, 0, player)
35 $enemy << Bomber::EnemyTrace.new( 7, 3, 0, player)
36 $enemy << Bomber::EnemyTrace.new( 7, 6, 0, player)
37 $enemy << Bomber::EnemyTrace.new( 8, 9, 0, player)
38 $enemy << Bomber::EnemyTrace.new( 13, 1, 0, player)
39 $enemy << Bomber::EnemyTrace.new( 14, 3, 0, player)
40 $enemy << Bomber::EnemyShooter.new(6, 10, 0, 0.3)
41 $enemy << Bomber::EnemyShooter.new(10, 4, 0, 0.3)
42 $enemy << Bomber::EnemyWizard.new(13, 9, 0)
43 $enemy << Bomber::EnemyWizard.new(15, 13, 0)
44 $enemy << Bomber::EnemyHammer.new(10, 3, 0)
45 $enemy << Bomber::EnemyHammer.new(10, 13, 0)
46
47 $enemy.each do |enel|
```

プログラムを保存したら、コマンドプロンプトを開いて、以下のコマンドを入力し、エンターキーを押してみましょう。

ruby main.rb

以下のようなウィンドウが開いて、敵が増えていることが確認できれば、ステップ7は終了です。



ウィンドウが開かない場合はプログラムが間違っているので見直しましょう。
コマンドプロンプトにエラーが出ているかもしれないの、確認してみましょう。

ステップ8.得点を表示する

bomber/statusbar.rb

ステータスバーに残りの敵の数を表示する機能を作ります。

エディタで新規作成を選択して、新しいファイルを作ります。

以下のようにプログラムを入力し、bomberディレクトリに『statusbar.rb』として保存します。

```
1 module Bomber
2   class Statusbar < Bomber::Character
3     def enemy_status(num)
4       font = new_font(16)
5       width = BLOCK
6       height = BLOCK
7       image = Image.new(width, height)
8       image.draw_font(0, (font.size + 1), "×#{num}", font, [0, 0, 0])
9       @font = Sprite.new(x+BLOCK, y-BLOCK/4, image)
10      @font.draw
11    end
12  end
13 end
```

bomber/score.rb

ステータスバーにスコアを表示する機能を作ります。

エディタで新規作成を選択して、新しいファイルを作ります。

以下のようにプログラムを入力し、bomberディレクトリに『score.rb』として保存します。

```
1 module Bomber
2   class Score < Bomber::Character
3     def score_status
4       font = new_font(20)
5       width = BLOCK*20
6       height = BLOCK*2
7       image = Image.new(width, height)
8       image.draw_font(0, (font.size + 1), "得点 #{$score} 点", font, [0, 0, 0])
9       @font = Sprite.new(x+BLOCK, y-BLOCK/2, image)
10      @font.draw
11    end
12  end
13 end
```

bomber/timer.rb

ステータスバーに残り時間を表示する機能を作ります。

エディタで新規作成を選択して、新しいファイルを作ります。

以下のようにプログラムを入力し、bomberディレクトリに『timer.rb』として保存します。

```
1 module Bomber
2   class Timer < Bomber::Character
3     def initialize(costume, x, y, angle, timelimit)
4       @timelimit = timelimit
5       @start_at = Time.now
6       super(costume, x, y, angle)
7     end
8
9     def time_status
10    time = (Time.now - @start_at).to_i
11    font = new_font(20)
12    width = BLOCK*20
13    height = BLOCK*2
14    image = Image.new(width, height)
15    image.draw_font(0, (font.size + 1), "残り時間 #{@timelimit - time} 秒", font, [0, 0, 0])
16    @font = Sprite.new(x+BLOCK, y-BLOCK/2, image)
17    @font.draw
18    if @timelimit - time < 0
19      gameover = Bomber::GameOver.new
20      gameover.z = 10
21      sleep 3
22      exit
23    end
24  end
25 end
26 end
```

main.rb

ステータスバーに上記で作成した機能を配置する処理を追加します。

『main.rb』をエディタで開いて、以下のようなプログラムをファイルの最後に追加してください。

```
90
91 statusbar1 = Bomber::Statusbar.new("../image/ene.png", 0, 15, 0)
92 statusbar1.on(:start) do
93   loop do
94     self.enemy_status(normal_enemy_count)
95   end
96 end
97
98 statusbar2 = Bomber::Statusbar.new("../image/ene2.png", 2, 15, 0)
99 statusbar2.on(:start) do
100   loop do
101     self.enemy_status(trace_enemy_count)
102   end
103 end
104
105 statusbar3 = Bomber::Statusbar.new("../image/wiz.png", 4, 15, 0)
106 statusbar3.on(:start) do
107   loop do
```

```

108   self.enemy_status(wizard_enemy_count)
109 end
110 end
111
112 statusbar4 = Bomber::Statusbar.new("../image/hun.png", 6, 15, 0)
113 statusbar4.on(:start) do
114   loop do
115     self.enemy_status(hammer_enemy_count)
116   end
117 end
118
119 statusbar5 = Bomber::Statusbar.new("../image/bow.png", 8, 15, 0)
120 statusbar5.on(:start) do
121   loop do
122     self.enemy_status(shooter_enemy_count)
123   end
124 end
125
126 timer = Bomber::Timer.new("../image/clear.png", 10, 15, 0, 30)
127 timer.on(:start) do
128   loop do
129     self.time_status
130   end
131 end
132
133 score = Bomber::Score.new("../image/clear.png", 16, 15, 0)
134 score.on(:start) do
135   loop do
136     self.score_status
137   end
138 end

```

プログラムを保存したら、コマンドプロンプトを開いて、以下のコマンドを入力し、エンターキーを押してみましょう。

`ruby main.rb`

以下のようなウィンドウが開いて、ステータスバーが正常に表示されることが確認できれば、ステップ8は終了です。



ウィンドウが開かない場合はプログラムが間違っているので見直しましょう。
コマンドプロンプトにエラーが出ているかもしれないの、確認してみましょう。

ステップ9. ブロックの設定画面

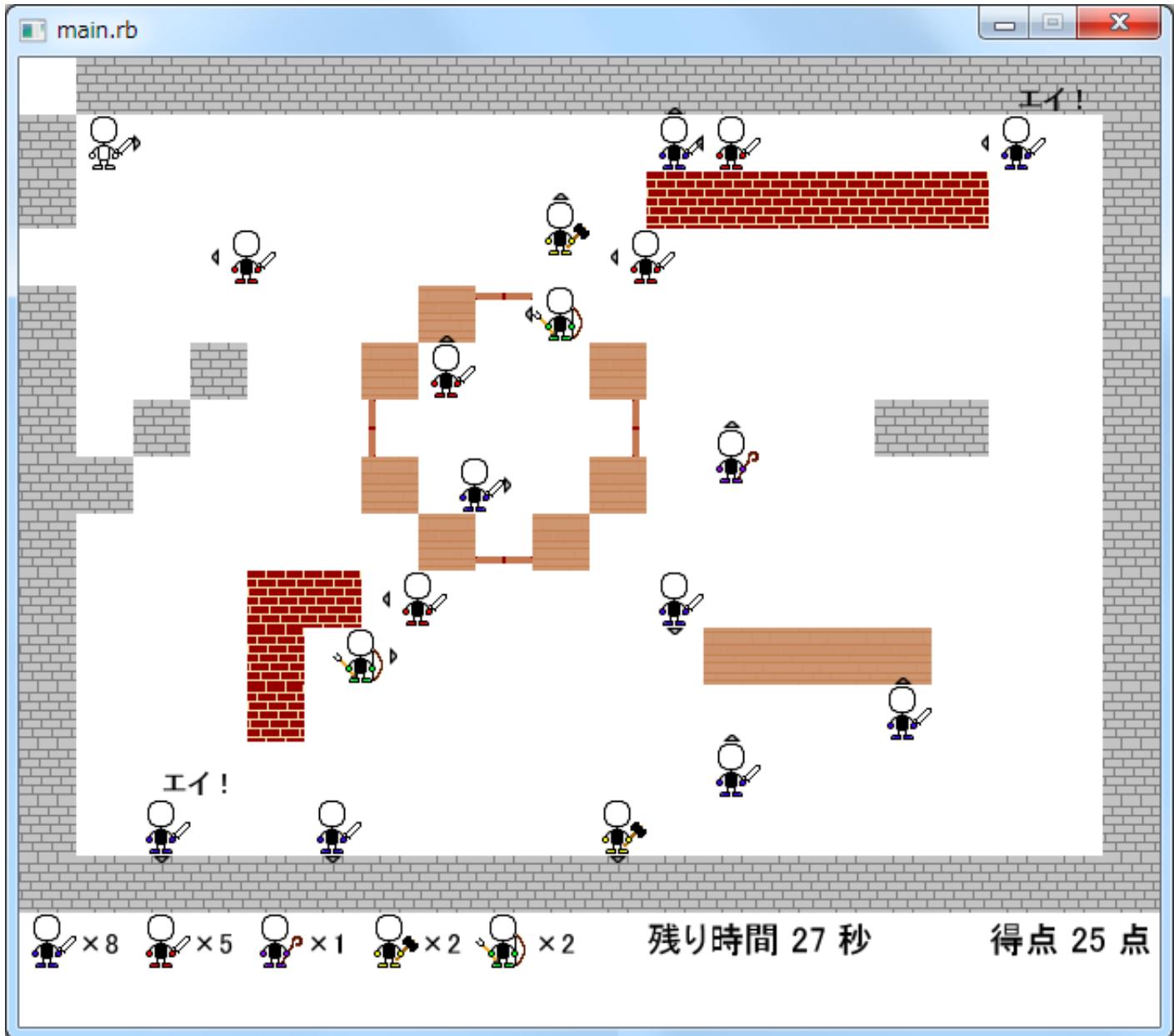
bomber/back.rb

config.rb

プログラムを保存したら、コマンドプロンプトを開いて、以下のコマンドを入力し、エンターキーを押してみましょう。

ruby main.rb

以下のようなウィンドウが開けば、ステップ9は終了です。



ウィンドウが開かない場合はプログラムが間違っているので見直しましょう。
コマンドプロンプトにエラーが出ているかもしれないで、確認してみましょう。

ここまででゲームの作成は完了です。

