最終グループレポート

6119019207 矢野大暉

2020/2/10 提出

1　ゲーム内容，操作方法（マニュアル）

1.1　ゲームのタイトル

AGENT 3

1.2　ジャンル

脱出ゲーム

1.3　概略

ゲームの設定は,「犯罪組織に盗まれた金塊を組織に潜入して，複数のプレイヤーと協力して，組織的に取り返す」といった設定である．本ゲームは，同一のサーバに接続した複数のプレイヤーが協力する脱出ゲームである．プレイヤーは出入り口からスタートし，敵 (NPC)，監視カメラに見からないように，ステージ上に設置された金塊をゲットし再び敵，監視カメラに見つからないように出入り口に帰ってくるゲームである．ステージは全部で 3 つ用意されており，ステージが進むごとに難易度を向上させる．

プレイヤーは敵キャラや監視カメラに対して妨害を行うことで，他プレイヤーが金塊を取るためのアシストを行うことができる．プレイヤーが行うことができる妨害として以下が挙げられる．

- 敵キャラに話しかける →　敵の視界の一時的な固定
- 監視カメラのハッキング →　監視カメラが一時的に停止
- 敵キャラに催涙スプレー →　敵の視界が一時的に無効化

1.4　プレイ人数

3 人

1.5　ルール

ルールは以下のようになる．
１．プレイヤー 3 人が入口から出発
２．プレイヤーは，ステージに配置されている金塊を奪い返しに行く
３．ステージに配置されている敵 (NPC)，監視カメラに見つからないように出口から脱出する

1.6　遊び方

以下の使用可能ツールを用いて，金塊を取り返し，出口からの脱出を試みる．

１．催涙スプレー
敵の視界を奪い，動きを止めることができる
(右下の自分のゲージがなくなるまで使用可能)
２．ハッキング
カメラの動きを止めることができる，各プレイヤーで 1 回のみ使用可能
(ゲージがたまるまでボタンを長押し，ゲージが溜まった状態で，ボタンを離した瞬間に 3 秒間カメラの首振りが停止する)
３．会話
敵の視界が固定できる
各 NPC に対して，計 3 回のみ使用可能

1.7　操作方法

本ゲームはゲームパッドでの操作を行う．

図 1: 操作方法

2　最終的なデータ構造，モジュール

2.1　サーバ，クライアント共通の変数，定数，構造体

ネットワーク関連の定数	
DEFAULT_PORT	デフォルトのポート番号 51000
MAX_LEN_NAME	クライアントのユーザ名の最大長
MAX_NUM_CLIENTS	接続要求の受付最大数
MAX_LEN_BUFFER	メッセージの最大文字数
MAX_LEN_ADDR	最大のアドレスの長さ
BROADCAST	ブロードキャスト

サーバーとクライアント間で送信されるコマンド	
MESSAGE_COMMAND	メッセージの送信
ZAHYO_COMMAND	座標の送信
KINKAI_COMMAND	金塊の状態
HACK_COMMAND	ハッキング状態
HACK_START_COMMAND	ハッキング開始
NOT_HACK_COMMAND	ハッキングキャンセル
PLAYER_COMMAND	プレイヤーが死んだことを示す
UP_COMMAND	上へのスティック操作が行われたことを示す
DOWN_COMMAND	下へのスティック操作が行われたことを示す
RIGHT_COMMAND	右へのスティック操作が行われたことを示す
LEFT_COMMAND	左へのスティック操作が行われたことを示す
CENTER_COMMAND	真ん中 (左右) へのスティック操作が行われたことを示す
AENTER_COMMAND	真ん中 (上下) へのスティック操作が行われたことを示す
X_ON_COMMAND	2 ボタンが押されたときに送信されるコマンド
X_OFF_COMMAND	2 ボタンが離されたときに送信されるコマンド
QUIT_COMMAND	ゲームの終了
START_COMMAND	メニュー画面で，スタートボタンを押したことを表す
ENEMY_MODIFY_COMMAND	敵 (NPC) の座標を同期する
TALK_START_COMMAND	会話を開始する
TALK_END_COMMAND	会話を終了する

CLIENT – クライアントの構造体		
データ型	変数名	内容
int	cid	クライアント ID
int	sock	ソケット番号
sockaddr_in	addr	IP アドレスやポート番号の情報
char[MAX_LEN_NAME]	name	名前

CONTAINER – コンテナの構造体		
データ型	変数名	内容
int	cid	クライアント ID
char	command	コマンド
char[MAX_LEN_BUFFER]	message	メッセージ
int	zahyo_x	x 座標
int	zahyo_y	y 座標
int[5]	enemy_zahyo_x	敵の x 座標
int[5]	enemy_zahyo_y	敵の y 座標
int[5]	move_angle	動く方向の角度
int[5]	prev_angle	現在向いている角度

2.2 クライアントに必要な変数，構造体

定数名	内容
WINDOWWIDTH	ウィンドウの幅
WINDOWHEIGHT	ウィンドウの高さ
PLAYER_NUM	プレイヤーの数
PLAYER_SPEED	プレイヤーのスピード
CAMERA_NUM	監視カメラの数
BACKGROUND_NUM	背景画像の数
FONT_NUM	システムメッセージの数
ENEMY_SPEED	敵 (NPC) のスピード
KOTEI_OBJECT_NUM_MAX	マップに読み込めるオブジェクトの最大の数
MAP_CHIPSIZE	マップの 1 オブジェクトの大きさ
MAP_WIDTH	横にマップのオブジェクトがいくつ置けるか
MAP_HEIGHT	縦にマップのオブジェクトをいくつ置けるか
SPRAY_WIDTH	催涙スプレーの幅
SPRAY_HEIGHT	催涙スプレーの高さ
SAIRUI_TIME	催涙スプレーで止まる時間
SPRAY_TIME	催涙スプレーが使える時間
TALK_TIME	会話時間
TALK_NUM	敵 (NPC) に何回まで話せるか
HACKTIME	ハッキングに要する時間
STOPTIME	ハッキング中に，カメラを止める時間
MENUMODE, GAMEMODE, RESULTMODE, STAGENUMMODE	ゲーム中のモード番号
ENEMY_NUM	敵 (NPC) の数

データ型	変数名	内容
const int	fps	1 秒間に何回画面を描画するか
const int	framedelay	1 回の描画にかけるべき時間
Uint32	framestart	フレーム処理の始まりの時間を格納する変数
Uint32	modi_before	プレイヤーの座標を修正するのに使う時間
int	frametime	1 回の処理にかかった時間を格納する
int	modi_time	座標更新の間隔の時間
u_short	port	ポート番号
char	server_name	サーバー名
TTF_Font*	japanesefont	ゲーム内フォント
bool	up	メニュー画面の上を選択していることを判別するフラグ
bool	down	メニュー画面の下を選択していることを判別するフラグ
bool	kinkai_flag	金塊描画フラグ
bool	hacking_flag	ハッキング処理フラグ
bool	player_flag	プレイヤー描画のフラグ
int	before_enemy_x	敵の以前の x 座標
int	before_enemy_y	敵の以前の y 座標
Uint32	random_start	ランダム移動を開始した時間を格納する変数
int	time_now	ハッキング時間を判定するときに判別する変数
float	gauge	ハッキングの際に描画されるゲージ
int[15][20]	map0,map1,map2	ゲームのマップの変数
static char *[TYPE_NUM]	imgfiles	読み込む画像ファイル
static char *[FONT_NUM]	fonts	システムメッセージの内容
static char *[2]	text_fukidashi	会話コマンドでフキダシの中に描画される内容
static SDL_Rect	camera_dst_rects	ステージごとのカメラの座標
static int [ENEMY_NUM]	enemy_moveangles	敵 (NPC) が最初に向いている方向
static enemymovetype [ENEMY_NUM]	enemy_movetypes	敵 (NPC) の動きのパターン
static SDL_Rect [FONT_NUM]	font_dst_rects	メッセージが描画される座標
int	kotei_object_num	1 マップのオブジェクトの数
int[ENEMY_NUM]	savestopenemy	動かないタイプの敵 (NPC) が何番目かを格納する変数
int	lrflag	敵の視界が左回りか右回りかを分けるフラグ

データ型	変数名	内容
int	status	ゲームの現在の状態
bool	run	ゲームが動作中かどうか
bool[ENEMY_NUM]	same_place_flag	敵 (NPC) が一定時間同じ座標にとどまっているかどうかを示すフラグ
bool[ENEMY_NUM]	random_start_flag	敵 (NPC) が一定時間同じ座標にとどまったため，ランダムウォークを始めたことを示すフラグ
Uint32[ENEMY_NUM]	stay_start	敵 (NPC) がとどまっている時間を格納する変数
int[ENEMY_NUM]	stay_time	敵 (NPC) がランダムウォークをする時間を格納する変数
int	myid	自分自身のクライアント ID を示す変数
int	stage_num	ステージの通し番号
bool	stage_trans_flag	ステージ遷移するか判別するフラグ
bool	game_over_flag	ゲームオーバーかどうか判別するフラグ

objectinfo – 画面に表示するオブジェクト (壁，敵など) 情報の構造体		
データ型	変数名	内容
objecttype	type	オブジェクトの種類
SDL_Texture*	image_texture	オブジェクトのテクスチャ
SDL_Rect	src_rect	元画像を読み取る領域
SDL_Rect	dst_rect	画像の出力先の領域

objectinfo – オブジェクトタイプを示す定数	
定数名	内容
TYPE_NONE	なにもないことを表す
TYPE_KINKAI	金塊を表す
TYPE_SHELF	棚を表す
TYPE_CAMERA	カメラを表す
TYPE_ENTRANCE	出入口を表す
TYPE_ENEMY	敵 (NPC) を表す
TYPE_PLAYER1	プレイヤー 1 を表す
TYPE_PLAYER2	プレイヤー 2 を表す
TYPE_PLAYER	プレイヤー 3 を表す
TYPE_ENEMY_MOVING_FLOOR_UL	移動床 1 を表す
TYPE_ENEMY_MOVING_FLOOR_UR	移動床 1 を表す
TYPE_ENEMY_MOVING_FLOOR_DL	移動床 1 を表す
TYPE_ENEMY_MOVING_FLOOR_DR	移動床 1 を表す
TYPE_ENEMY_MOVING_FLOOR_REV	移動床 1 を表す
TYPE_BACKGROUND	背景を表す
TYPE_SPRAY	催涙スプレーを表す
TYPE_NUM	タイプの総数を表す

inputkeys – ジョイパッドからの入力を保存する構造体		
データ型	変数名	内容
Uint32	left,right,up,down	アナログスティックの入力方法を保存する
Uint32	a,x,y,b	ボタンの入力方法を保存する

プレイヤーの構造体		
データ型	変数名	内容
objecttype	type	オブジェクトの種類
SDL_Texture*	image_texture	プレイヤーのテクスチャ
SDL_Texture*	spray_texture	催涙スプレーのテクスチャ
SDL_Rect	src_rect	元画像を読み取る領域
SDL_Rect	dst_rect	画像の出力先の領域
float	back_zahyo_x	小数で表されたプレイヤーの正確な x 座標
float	back_zahyo_y	小数で表されたプレイヤーの正確な y 座標
bool	flag_kinkai	プレイヤーが金塊を取得したか判別するフラグ
bool	flag_hack_start	ハッキングを開始フラグ
bool	flag_hack_end	ハッキングを終了フラグ
int	hack	ハッキングできる回数
int	inputtime	ハッキング開始した時間を保存する
int	speed	プレイヤーの速度
inputkeys	key	入力されたキー情報を保存
int	look_angle	プレイヤーが向いている角度
int	spray_flag	スプレーを出しているか判別するフラグ
SDL_Rect	spray_src_rect	催涙スプレーテクスチャを読み取る範囲
SDL_Rect	spray_dst_rect	催涙スプレーテクスチャを出力する範囲
int[2][4]	spray_hitline	催涙スプレーの当たり判定
SDL_Point	spray_origin	催涙スプレーが出てくる座標
int	spraytime	催涙スプレーが使える残り時間
int	talkstarttime	会話を開始した時間
bool	flag_talk	会話をしているか判別するフラグ
bool	flag_fukidasiflip	位置によって会話の際のフキダシを反転する必要があるか判定するフラグ

camerainfo – カメラ情報の構造体		
データ型	変数名	内容
SDL_Texture*	image_texture	カメラのテクスチャ
SDL_Rect	src_rect	元画像を読み取る領域
SDL_Rect	dst_rect	画像の出力先の領域
bool	flag_kinkai	金塊を持っているか判別するフラグ
bool	flag_hack	ハッキングをしているか判別するフラグ
int[2][3]	tri	カメラ視界の当たり判定の三角形
float[3]	theta	カメラの角度
clockwise	bool	カメラの回転方向を指定する
double	angle	カメラの向いている方向

enemyinfo – 敵 (NPC) の構造体		
データ型	変数名	内容
objecttype	type	オブジェクトの種類
SDL_Texture*	image_texture	敵 (NPC) のテクスチャ
SDL_Rect	src_rect	元画像を読み取る領域
SDL_Rect	dst_rect	画像の出力先の領域
SDL_Rect	prev_overlap_rect	一時的にオブジェクトと重なった範囲を保存する
bool	flag_kinkai	金塊を取ったかどうか判別するフラグ
bool	speed	敵 (NPC) の移動速度
int	move_angle	敵が動く方向
bool	flag_sairui	催涙スプレーを食らっているか判別するフラグ
enemymovetype	movetype	敵の動きのタイプ
unsigned int	savetime	催涙スプレーを食らった時間を保存する
int[2][3]	tri	敵 (NPC) 視界の当たり判定の三角形
int	prev_angle	現在敵 (NPC) が向いている角度
int	talk_angle	プレイヤーが会話してきたときの角度
bool	flag_talk	会話しているか判別するフラグ
int	talkstarttime	会話の開始時間
bool	flag_one_talk	会話を一回したか判別するフラグ
int	talktime	会話時間
int	talknum	これまでに会話した回数を保存
bool	flag_fukidasiflip	フキダシを反転するか判別するフラグ

backgroundinfo – 背景の構造体		
データ型	変数名	内容
SDL_Texture*	image_texture	背景のテクスチャ
SDL_Rect	src_rect	元画像を読み取る領域
SDL_Rect	dst_rect	画像の出力先の領域

fontinfo – フォントの構造体		
データ型	変数名	内容
SDL_Texture*	image_texture	フォントのテクスチャ
SDL_Rect	src_rect	元画像を読み取る領域
SDL_Rect	dst_rect	画像の出力先の領域

fukidashiinfo– フキダシの構造体		
データ型	変数名	内容
SDL_Texture*	image_texture	フキダシのテクスチャ
SDL_Rect	src_rect	元画像を読み取る領域
SDL_Rect	dst_rect	画像の出力先の領域
SDL_Texture*[2]	textimage	会話時のテキスト内容
SDL_Rect	font_src_rect	フォントの元を読み取る領域

2.3 クライアントで使用するモジュール

void Startup(void)	
関数名	Startup
機能	ゲームの初期化を行う関数
引数	なし
返り値	なし

void SetCamera(void)	
関数名	SetCamera
機能	カメラの初期値を設定する関数
引数	なし
返り値	なし

void Fontload(void)	
関数名	Fontload
機能	フォントの読み込みを行う関数
引数	なし
返り値	なし

void Imageload(void)	
関数名	Imageload
機能	背景，カメラ画像の読み込みを行う関数
引数	なし
返り値	なし

void MakeMap(void)	
関数名	MakeMap
機能	マップの読み込みと配置を行う関数
引数	なし
返り値	なし

void InitObjectFromMap(void)	
関数名	InitObjectFromMap
機能	マップ変数からオブジェクトの配置と初期化を行う
引数	なし
返り値	なし

void setup_client(cahr *server_name ,u_short port)	
関数名	setup_client
機能	クライアントのセットアップを行う関数
引数	なし
返り値	なし

void send_data(void *data, int size	
関数名	send_data
機能	サーバにデータを送信する関数
引数	なし
返り値	なし

void Input(void)	
関数名	Input
機能	ジョイパッドの入力を読み取る関数
引数	なし
返り値	なし

int control_requests(void)	
関数名	control_requests
機能	サーバからのデータを受信し，データに応じた処理をする関数
引数	なし
返り値	0 or 1

int execute_command(void)	
関数名	execute_command
機能	サーバから送られてきたコマンドを実行する
引数	なし
返り値	0 or 1

void receive_data(void *data, int size)	
関数名	receive_data
機能	サーバからのデータを受け取る関数
引数	なし
返り値	なし

void Stage_Renew(void)	
関数名	Stage_Renew
機能	ステージ遷移する際にステージの初期化をする関数
引数	なし
返り値	なし

void DrawMenu(void)	
関数名	DrawMenu
機能	メニュー画面を描画する
引数	なし
返り値	なし

void MoveChara(void)	
関数名	MoveChara
機能	キャラクター (敵，プレイヤーなど) を動かす関数
引数	なし
返り値	なし

void PlayerAction(void)	
関数名	PlayerAction
機能	催涙スプレー，会話処理を行う関数
引数	なし
返り値	なし

void ChangeEnemyMoveAngle(void)	
関数名	ChangeEnemyMoveAngle
機能	敵 (NPC) があるオブジェクトに重なったときに特定の方向へ動く方向を変える関数
引数	なし
返り値	なし

void MoveTriangle(void)	
関数名	MoveTriangle
機能	カメラ，敵の視界の三角形を動かす
引数	なし
返り値	なし

float Rotation(int x1, int y1, int a, int b, double theta, int *x2, int *y2)	
関数名	Rotation
機能	x と y と角度を引数に与えて回転後の座標を返す関数
引数	x 座標，y 座標，角度
返り値	回転後の座標

void Collision(void)	
関数名	Collision
機能	カメラ，敵 (NPC) との衝突を検知する関数
引数	なし
返り値	なし

void RenderWindow(void)	
関数名	RenderWindow
機能	画面の描画を行う関数
引数	なし
返り値	なし

void Events(void)	
関数名	Events
機能	ハッキングを行う関数
引数	なし
返り値	なし

void StageNumShow(void)	
関数名	StageNumShow
機能	ステージに応じた表示をする関数
引数	なし
返り値	なし

void terminate_client(void)	
関数名	terminate_client
機能	ソケットの切断を行う関数
引数	なし
返り値	なし

void joystick_send(int num)	
関数名	joystick_send
機能	ジョイスティックの操作に関する情報を送信する関数
引数	コマンド番号
返り値	なし

void handle_error(void)	
関数名	handle_error
機能	エラーメッセージを出力する関数
引数	なし
返り値	なし

void Destroy(void)	
関数名	Destroy
機能	ゲームを終了する関数
引数	なし
返り値	なし

2.4 サーバに必要な変数，構造体

データ型	変数名	内容
int	num_clients	クライアント数
int	num_socks	ソケット数
fd_set	mask;	FD 集合を表す構造体
int	start_count	スタートカウント
int[3]	start_count_flag	クライアント 3 台の同期をとるためのフラグ

2.5 サーバで使用するモジュール

void setup_server(int num_cl, u_short port)	
関数名	setup_server
機能	サーバーのセットアップを行う関数
引数	クライアント数, ポート番号
返り値	なし

int control_request(void)	
関数名	control_request
機能	クライアントからのデータを受信し，データに応じた処理をする関数
引数	なし
返り値	1 のとき処理を続け，0 のとき処理を終了する．

void terminate_server(void)	
関数名	terminate_server
機能	サーバを終了する処理
引数	なし
返り値	なし

void send_data(int cid, void *data, int size)	
関数名	send_data
機能	クライアントにデータを送信する関数
引数	クライアント ID, データ, データサイズ
返り値	なし

void receive_data(int cid, void *data, int size)	
関数名	send_data
機能	クライアントからデータを受信する関数
引数	クライアント ID, データ, データサイズ
返り値	なし

void handle_error(void)	
関数名	handle_error
機能	エラーメッセージを出力する関数
引数	なし
返り値	なし

3　ファイルの構成, プログラムの起動方法, 注意点

common.h クライアントで使用するヘッダファイル
constants.h サーバ，クライアントで使用するヘッダファイル

```
── source クライアント側のファイルをまとめたフォルダ
 │ agent クライアントの実行ファイル
 │ fonts-japanese-gothic.ttf 日本語のフォントファイル
 │ func.c クライアントで使用する関数をまとめたファイル
 │ func.h func.c で使用するヘッダファイル
 │ main.c クライアントのメインプログラム
 │ Makefile クライアントの make ファイル
 │
 ├── bgm ゲーム内で使用する BGM
 │ bgm1.mp3
 │
 └── images ゲーム内で使用する画像素材
        bg.png
        camera.png
        enemy.png
```

entrance.png
floor.png
floor_dl.png
floor_dr.png
floor_rev.png
floor_ul.png
floor_ur.png
fukidasi.png
fukidasi.xcf
kinkai.png
menu.png
player.png
player1_8pattern.png
player1_8pattern.xcf
player2.png
player2_8pattern.png
player2_8pattern.xcf
player3.png
player3_8pattern.png
player3_8pattern.xcf
shelf.png
spray.png

— source_server サーバ側のファイルをまとめたフォルダ
main.c サーバのメインプログラム
main.o
Makefile サーバの make ファイル
server サーバの実行ファイル
server.c サーバで使用する関数をまとめたファイル
server.o

3.1 プログラムの起動，終了方法

3.1.1 起動方法

1．プレイヤー 3 人のうちの誰かがサーバを起動する．
./server 3
2．各プレイヤーがサーバに繋げる
./agent clpc108

3.1.2　終了方法

ゲームパッドの 12 番ボタンを押す.

4　班としての反省点

ゲームの作り込みを優先するあまり，ゲームの基礎部分がおろそかになってしまい実装が遅れた点．プレイヤーを追跡する NPC の動きやプレイヤーの妨害アクションの実装にこだわり，ステージ遷移とゲーム終了処理の実装を後にまわしてしまった．最終的にはこれらも完成させることができたが，スケジュールとしてはギリギリなものになった．それ以外の開発についてはスケジュールどおりに進めることができた．