オペレーティングシステム演習

T23CS015 菊池零音

1. 作成プログラムの概要

サーバ側とクライアント側で鬼ごっこを行い.サーバ側が鬼役とし,クライアント側が逃げる役としてプレイする.
マップには障害物「#」があり,これを超えて進むことはできない.
また,マップには飛び超えることのできる障害物「+」がある.飛び越える方向の矢印キーを押すことで先に飛び越えることができる.(通常の移動キーでは飛び越えることができない)
ウィンドウは二つありマップにある「W」から別ウィンドウへワープすることができる.
サーバがクライアントを捕まえたらウィンドウが変わり勝敗をお知らせする.
また,途中で Q キーを押すことでゲームを終了することができる.

2. 操作方法

I キー:　　上方向に移動
J キー:　　左方向に移動
K キー:　　下方向に移動
L キー:　　右方向に移動
↑キー:　　上方向の飛び越えられる壁を越えて移動
←キー:　　上方向の飛び越えられる壁を越えて移動
↓キー:　　上方向の飛び越えられる壁を越えて移動
→キー:　　上方向の飛び越えられる壁を越えて移動
Q キー:　　ゲーム終了

3. プログラムの内部構造の概要(☆は自作した部分)

TagGame* initTagGame(char myChara, int mySX, int mySY,
　　char itChara, int itSX, int itSY)
鬼ごっこゲームの初期化
☆ゲームの位置座標やプレイヤーのいるマップの初期化
☆メインウィンドウとサブウィンドウの表示
・画面の初期化

void setupTagGame(TagGame *game, int s)
鬼ごっこゲームの準備
・データ入力のための準備

オペレーティングシステム演習

T23CS015 菊池零音

☆ウィンドウに表示する障害物などのマップをここで生成し,描画する

void playServerTagGame(TagGame *game)
サーバ側の鬼ごっこゲームの開始
・入力キーの読み込み,プレイヤーの状態の更新・表示,クライアントに情報送信
☆鬼が追いついたときの処理
☆Q キーを押されたときの処理

void playClientTagGame(TagGame *game)
クライアント側の鬼ごっこゲームの開始
・入力キーの読み込み,ゲームの状態更新・表示,サーバに入力キー情報を送信
☆逃げる役が追いつかれた時の処理
☆Q キーを押されたときの処理

void destroyTagGame(TagGame *game)
鬼ごっこゲームの後始末
・ウィンドウの破棄やオブジェクトの解法

static void getServerInputData(TagGame *game, ServerInputData *serverData)
サーバ側でデータ届いているファイルでスクリプタからデータを読む
・データが届いているファイルデスクリプタを調べ届いているなら押したキー情報を抽出する

static void getClientInputData(TagGame *game, ClientInputData *clientData)
クライアント側でデータが届いているファイルでスクリプタからデータを読む
☆自分と相手の座標情報を抽出する際に自分と相手のマップの情報もやり取りする

static void updatePlayerStatus(TagGame *game, ServerInputData *serverData)
プレイヤーの情報を更新する
・前回のプレイヤー情報を保存しキーに応じて処理を行う
☆移動先に壁があったら移動キーを押しても移動しないに処理を行う
☆飛び越え可能な壁があったら矢印キーで飛び越えられるように処理を行う
☆移動先がワープポイントであったら別ウィンドウへワープするよう処理を行う

オペレーティングシステム演習

T23CS015 菊池零音

static void copyGameState(TagGame *game, ClientInputData *clientData)
ゲームの状態を更新する
・前回のプレイヤー情報を保存
☆自分と相手の座標とプレイヤーのいるマップを更新する.

static void printGame(TagGame *game)
ゲーム画面を表示する
・前回の位置を消去し今のプレイヤーを描画する
☆移動した後の前回の位置の消去を前回の自分と相手がいたマップを指定して処理を行う

static void sendGameInfo(TagGame *game)
ゲームの状態を相手に知らせる
・情報が更新されているなら新しい情報を送信する
☆プレイヤーの座標情報を送信する際にプレイヤーのいるマップの情報も送信する

static void sendMyPressedKey(TagGame *game, ClientInputData *clietData)
自分の押しているキーを相手に送る
・押されているなら自分の押しているキー情報を送る

static void die()
端末を復元し終了

以下は自作関数
void showText(TagGame *game,char *text,int WinX,int WinY,int penID)
ウィンドウにテキストを表示させる.
第 2 引数は,表示させるテキスト
第 3,4 引数は,テキストを表示させる座標
第 5 引数は,使用するペン

int** readLine(char *mapName)
第 1 引数で読み込むテキストファイル名を指定し.テキストファイルを読み込み, 2 次元配列のマップデータに落とし込む

void createMap(TagGame *game,WINDOW *Win,char *mapName)

第２引数で指定したウィンドウに,第３引数で指定したマップをウィンドウに表示させる

void warp(TagGame *game,Player *character)
第１引数でプレイヤーを指定したキャラクターをワープさせる

WINDOW* chooseWin(TagGame *game,Player *character)
int** chooseMap(TagGame *game,Player *character
int chooseMap_Lines(TagGame *game,Player *character)
int chooseMap_Colums(TagGame *game,Player *character)
それぞれ第２引数で指定したプレイヤーのキャラクターのいるウィンドウ,マップ,マップの縦の大きさ,マップの横の大きさを返す.

4. 工夫した点・アピールポイント

・はじめは ShowText 関数ではなくて ShowWin や ShowLose 関数などの特定の文字を表示する関数を複数作っていたが,引数で出力する文字を指定する関数にすることで,再利用性が高くしたこと.

・最後の結果表示をする際に,ウィンドウの背景色や文字の色を変えることでゲーム画面ではなく結果を出力していることがわかりやすいようにしたこと.また最後のウィンドウで同じ文字を出力するのではなく勝利,敗北,中断の 3 種類があり,勝利と敗北が表示されてから数秒後に「Thank you for playing!!」と文字が変わること.

・ただ障害物をよけて移動するだけでは面白くないので,いくつか超えることのできる障害物を作ったこと.

・現在のマップでは飛び越えられる障害物を飛び越えることで障害物にめり込んでしまうことはないが,マップが変わった時に,このようなことが起こらないように安定性の高いコードを書いたこと.

・中央集権型のプログラミングにすることでデータが遅れて届くことがないように一貫性管理をしている.

・できるだけ変数名や関数名を x や y などを使わないようにして,変数名や関数名でどのようなものなのかわかりやすいようにコードを書いたこと