GameStateFactory.c

/\* Create the game state for a new game \*/

GameState\* CreateInitialGameState()

{

GameState\* initialGameState; /\* the new GameState object we will return \*/

/\* create an empty GameState object \*/

initialGameState = GameState\_Create();

/\* the user starts with a score of 0 \*/

initialGameState->score = 0;

/\* the user starts with a count of 0 \*/

initialGameState->count = 0;

/\* the user starts in room 0 \*/

initialGameState->currentRoomIndex = 5;

/\* return the new object \*/

return initialGameState;

}

GameState.h

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

filename GameState.h

author Matthew Picioccio

DP email mattpic@digipen.edu

course GAM100 \*\* Do not use this code in your team project

Brief Description:

This file declares the GameState interface, which is used to reflect the user state

in the game.

The mutation functions are recommended because they provide a consistent

user experience for operations, though those operations could also be performed

directly on the object.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#pragma once

#include "stdafx.h" /\* bool \*/

typedef struct ItemList ItemList;

typedef struct GameFlags GameFlags;

/\* The user state in the game \*/

typedef struct GameState

{

/\* the user's score \*/

int score;

/\* the user's current turn count \*/

int count;

/\* the user's flags \*/

GameFlags\* gameFlags;

/\* the user's inventory \*/

ItemList\* inventory;

/\* The index of the room the user is currently in \*/

int currentRoomIndex;

/\* If false, the game is over and should exit \*/

bool isRunning;

} GameState;

/\* Create an empty game state object \*/

GameState\* GameState\_Create();

/\* Free the memory associated with a game-state object \*/

void GameState\_Free(GameState\*\* gameStatePtr);

/\* Modify the score, with appropriate user information \*/

void GameState\_ChangeScore(GameState\* gameState, int modifier);

/\* End the game, with appropriate user information \*/

void GameState\_EndGame(GameState\* gameState, const char\* message);

int GetCount(GameState\* gamestatePtr);

**TimeCommandHandler.c**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

filename TimeCommandHandler.c

author Andrew Yan

course GAM100 \*\* Do not use this code in your team project

Brief Description:

This file defines functions that handle the "time" user command, which outputs

a list of the available verbs.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "stdafx.h" /\* UNREFERENCED\_PARAMETER \*/

#include "TimeCommandHandler.h" /\* Function declarations \*/

#include "CommandList.h" /\* CommandList\_Print \*/

#include "CommandData.h" /\* struct CommandData \*/

#include "GameState.h"

/\* Handles the "time" command, which outputs a list of available verbs \*/

void HandleTimeCommand(CommandData \*command, GameState \*gameState, WorldData \*worldData)

{

/\* avoid W4 warnings on unused parameters - this function conforms to a function typedef \*/

UNREFERENCED\_PARAMETER(worldData);

printf("You have currently taken %i minutes\n", GetCount(gameState) );

}

**TimeCommandHandler.h**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

filename TimeCommandHandler.h

author Andrew Yan

DP email N/A

course GAM100 \*\* Do not use this code in your team project

Brief Description:

This file declares functions that handle the "time" user command, which calls

an item-specific function on a given item

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#pragma once

typedef struct CommandData CommandData;

typedef struct GameState GameState;

typedef struct WorldData WorldData;

/\* Handles the "break" command, which removes an item and adds an item \*/

void HandleTimeCommand(CommandData\* command, GameState \*gameState, WorldData \* worldData);

**GoCommandHandler.c**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

filename GoCommandHandler.c

author Matthew Picioccio

DP email mattpic@digipen.edu

course GAM100 \*\* Do not use this code in your team project

Brief Description:

This file defines functions that handle the "go" user command, which moves

the user from one room to another using defined exits.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "stdafx.h" /\* NULL, printf \*/

#include "GoCommandHandler.h" /\* Function declarations \*/

#include "CommandData.h" /\* struct CommandData \*/

#include "GameState.h" /\* struct GameState \*/

#include "WorldData.h" /\* WorldData\_GetRoom \*/

#include "Room.h" /\* Room\_GetNextRoomIndex, Room\_Print, INVALID\_DIRECTION\_ID \*/

/\* Handles the "go" command, which moves the user to another room \*/

void HandleGoCommand(CommandData \*command, GameState \*gameState, WorldData \*worldData)

{

Room\* currentRoom; /\* the room we are currently in \*/

int nextRoomIndex; /\* the index of hte next room \*/

/\* safety check on the parameters \*/

if ((command == NULL) || (command->noun == NULL) || (gameState == NULL) || (worldData == NULL))

{

return; /\* take no action if the parameters are invalid \*/

}

/\* get the current room, based on the user state \*/

currentRoom = WorldData\_GetRoom(worldData, gameState->currentRoomIndex);

/\* get the next room index, based on the noun used with "go" \*/

if (Room\_GetNextRoomIndex(currentRoom, command->noun, &nextRoomIndex) == false)

{

/\* there is no exit in that direction, so let the user know and take no other action \*/

printf("You cannot move %s.\n", command->noun);

return;

}

/\* update the game state to move to the new room \*/

gameState->currentRoomIndex = nextRoomIndex;

/\* output the successful action \*/

printf("You move %s.\n\n", command->noun);

/\* get the new room \*/

currentRoom = WorldData\_GetRoom(worldData, gameState->currentRoomIndex);

/\* print the description of the new room \*/

Room\_Print(currentRoom);

gameState->count += 1;

}