CUT THE POWER - GRAPHICS

Cut the Power: A Data Communications Project

Team members:

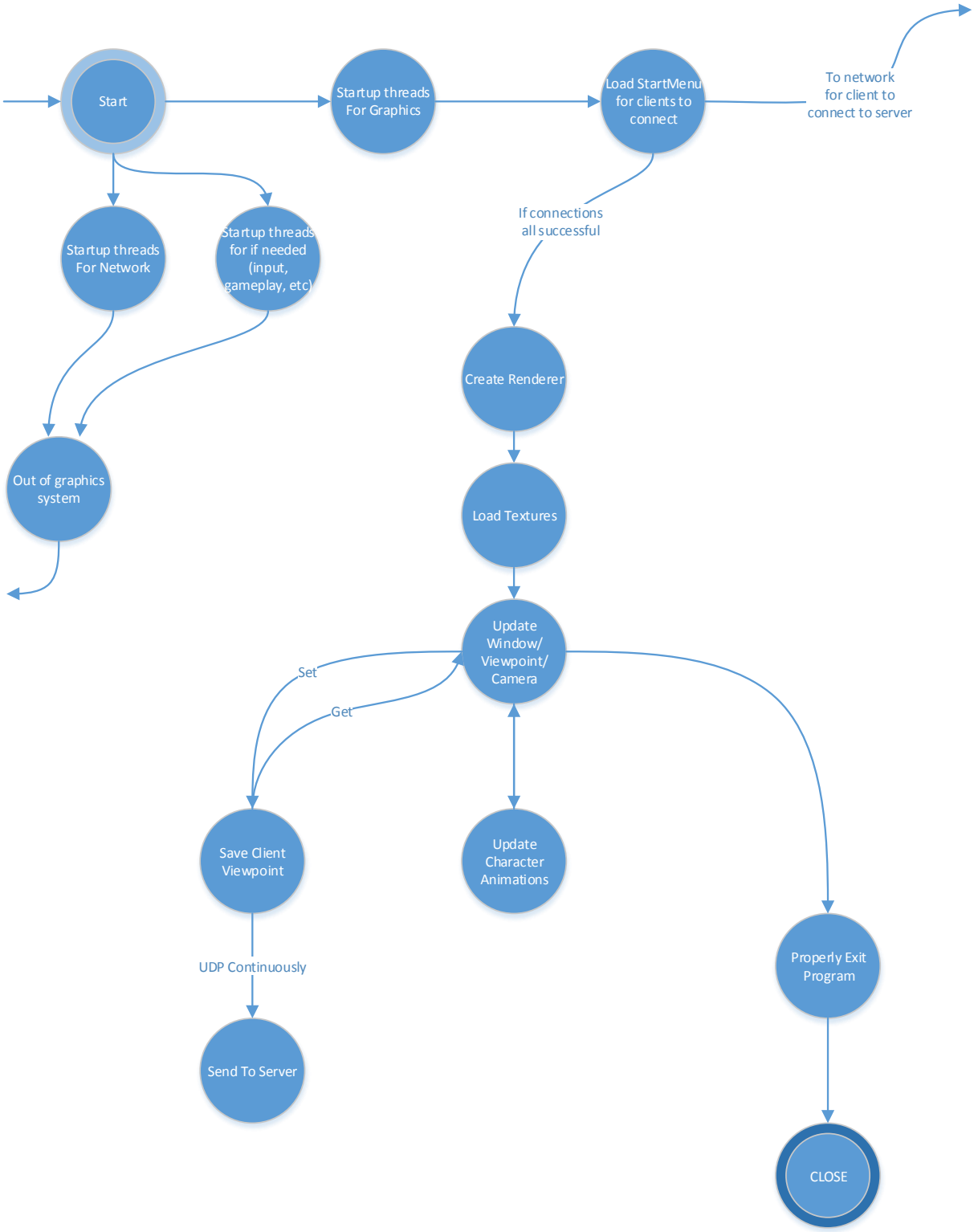
Robin Hsieh, Sam Youssef, Tim Kim

Konstantin Boyarinov, Damien Sathanielle, Mateusz Siwoski

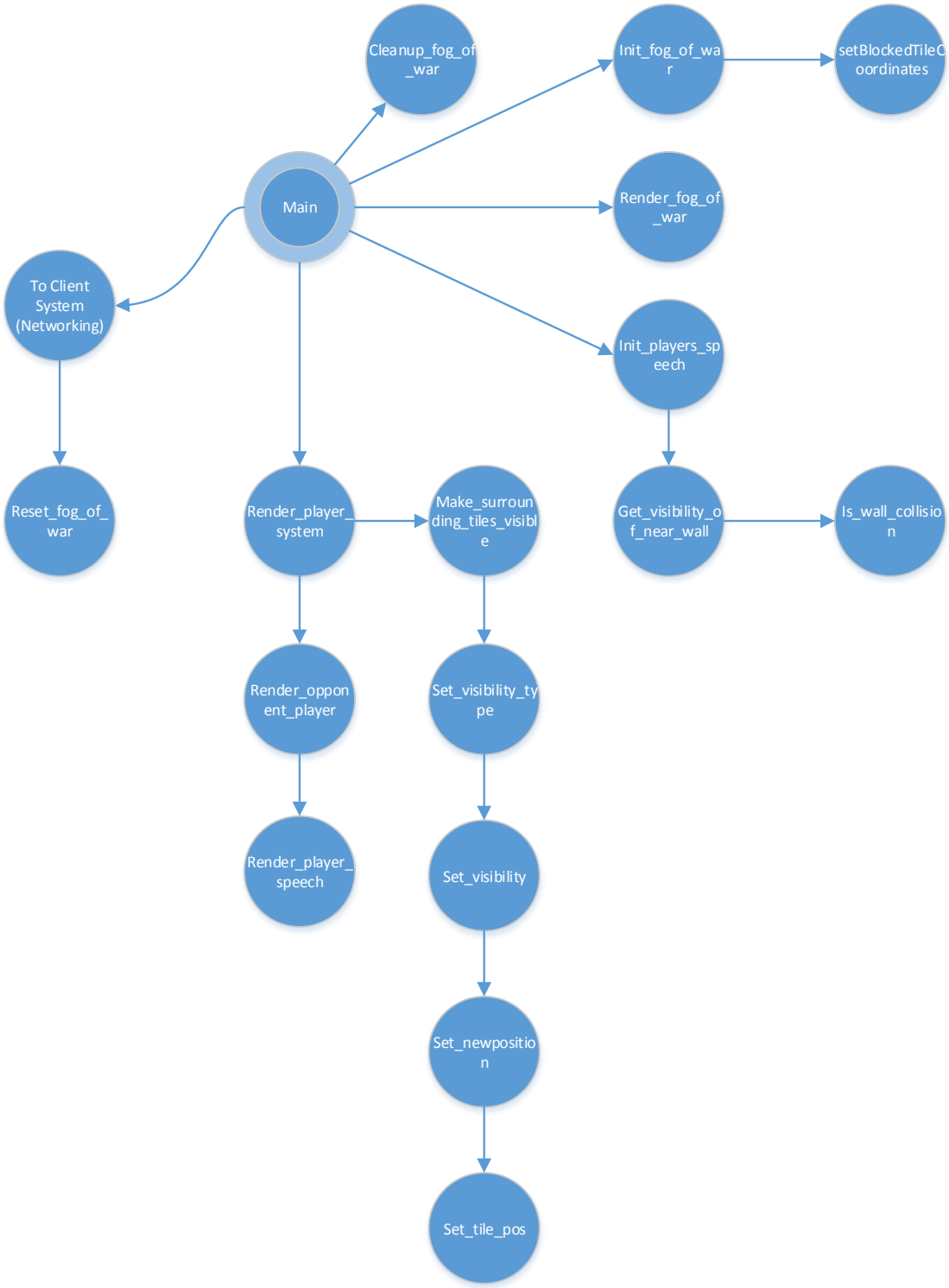
Table of Contents

State Diagram:

Overview of Graphics in general for player and map rendering:



Fog Of War and Players Speech Systems (By Sam Youssef)



Pseudo Code:

```
int main(){
	initialize(){}
	create thread for loading images()
	bool quit = false;
	SDL_Event e;//Event handler
	while( !quit )	//While application is running{
		while( SDL_PollEvent( &e ) != 0 )//Handle events on queue{
			if( e.type == SDL_QUIT ) //User requests quit.{
				quit = true;
			}
		}
		render_system(&world, surface);
		SDL_RenderClear( gRenderer ); //Clear screen
		SDL_RenderCopy( gRenderer, gTexture, NULL, NULL ); //Render texture to screen
		SDL_RenderPresent( gRenderer );//Update screen
		SDL_UpdateWindowSurface(window);
	}
	close();
}

bool initialize(){
	initialize SDL{
	}
	create window(){
	}
	create renderer(){
	}
	initialize img_loading{
	}
	return true;
}

bool load_media(){}

void close(){
	SDL_DestoryTexture();
	SDL_DestroyRenderer();
	SDL_DestroyWindow();
	free();
	IMG_QUIT();
	SDL_QUIT();
}
```

FOG OF WAR SYSTEM & PLAYER SPEECH SYSTEM

Note: the Player Speech System is an outgrowth of the Fog of War System. Speech is exchanged only when an opponent player is visible to your player.

FOG OF WAR SYSTEM

Sam Youssef

init_fog_of_war_system

Initializes and fills the following:

- a tile map array (containing the coordinates and the visibility level of each 40x40 (in pixels) tile on the map

- an array containing fully opaque 40x40 fog of war surfaces.
- an array containing 40x40 transparent fog of war sufaces
- calls setBlockedTileCoords

setBlockedTileCoords

Sets the coords of tiles that are blocked (within the player's circle of vision) if they lay between the player and a wall tile. This is done by filling a table indicating which other tiles are blocked if the tile in question is blocked.

render_fog_of_war_system

Loops through a tile map multidimensional array containing all tiles belonging to the map.

Each element of the tile map is a struct which contains a rectangle of the tile's coordinates and an integer value determining a tile's level of visibility: fully opaque, transparent or visibile to the team.

Blits (draws) the tile surface to the window surface using the tile's coordinates. The type of tile to blit (transparent, opaque, etc.) is determined by the visibility index.

Cops have transparent visibility set to default whereas robbers have opaque visibility set to default.

cleanup_fog_of_war

Frees all dynamically allocated arrays, including sound effects and blitting surfaces.

reset_fog_of_war

Resets the fog of war to opaque (i.e. resetting which tiles have been visited). This is done by looping through the tile map and setting each tile's visibility to opaque.

render_opponent_players

By default, the function render_player_system only renders you and friendly players.

To render opponents, this function loops through an array of enemy entities and renders each one if their coordinates match a visible tile you or your team control.

It creates an SDL_Rect based on the player's position and blits it to the display surface.

Finally, it calls render_player_speech to play a sound.

make_surrounding_tiles_visible

Starts by iterating through a 7 x 7 tile area (280 x 280 pixel) around the player and marking which tiles are not visible in a table.

This is determined by whether or not a tile is a wall tile. To determine if a tile is a wall tile, get_visibility_of_near_wall is called. If a wall tile is found, the array of tiles blocked by this tile is iterated through and all of those wall tiles are subsequently marked as not visible (given a value of 1 in the table).

Once the loop completes, the most immediate wall tiles need to be greyed out (rather than rendered as opaque) in order to create a more realistic fog of war.

Starting from the players position, eight loops are generated starting from the player's position in all eight directions around the player. These break once hitting the first wall tile encountered (which is then assigned a value of 2 in the table).

To complete the effect, a series of if statements do the same thing, but starting at the areas surrounding the corners (such as south-south-west, etc.)

get_visibility_of_near_wall

This is simply a wrapper function which calls is_wall_collision on a tile whose coordinates are passed in as arguments.

is_wall_collision

Loops through the array of entities found in the world struct to determine whether a particular tile is a wall. Returns COLLISION_WALL or COLLISION_EMPTY.

set_tile_visibility

Wrapper function for set_visibility. Determines which type of visibility is passed to set_visibility based on the tile's visibility index. e.g. Transparent for index of 2, clear for index of 0, etc.

set_visibility

Loops through the array of entities in the world struct to determine the level (floor) of the tile whose visibility is being set. Changes the visibility of the tile by setting the tile's visibility flag (for that particular level) to the visibility type (passed as an argument)

If the tile is in sight of the controllable player, writes the coordinates of the tile to a special array which holds those coordinates.

set_newposition

Creates a new PositionComponent struct from an old one passed in by reference and x and y coordinates representing deltas relative to the old PositionComponent's position.

Adds deltas to old x and y coordinates by calling set_tile_pos

set_tile_pos

Adds or subtracts two position "safely". That is, adds two values but makes sure that the new values cannot exceed the map's boundaries.

PLAYER SPEECH SYSTEM

Sam Youssef

init_player_speech

Loads sound files to arrays of sound files. One for each team (COPS or ROBBERS).

render_player_speech

If time since last time played exceeds eight seconds and if an opponent is visible on a tile YOU (not someone else on your team) control, play a random sound from the list of opponent sounds.

Milestones:

Milestone 1 – Simple Map

Due by **February 21, 2014**. This milestone will include the game's initialisation. The first iteration of the game will follow a very simplistic implementation of Cops vs. Robbers. The game will display the map on screen, render up to two players, and allow for user input to maneuver a player around the map. When the game begins, there will be a map that will be displayed with a perimeter wall to prevent users from going off-screen as well as some internal walls to impede players. Throughout the map, there will be 4 placed "objectives" that the Robber must compromise.

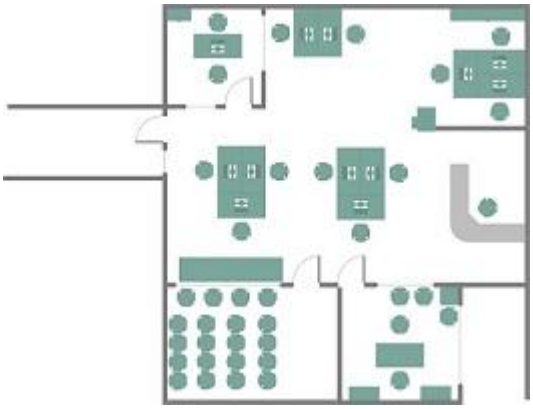


FIGURE 1: A SEGMENT OF A MAP FROM A USER'S POINT OF VIEW

FIGURE 2: OUR REFERENCE IN DESIGNING THE STYLE OF THE GAME

Milestone 2 – Better Maps/Menu

Due by **March 6, 2014** . The overall graphics and map details will be more polished from the first milestone.

Multiple floors will be introduced with a maximum of 8, which will be accessible through stairs. When a player enters a stairwell going up or down, they will be moved up or down a floor accordingly with the screen going blank. The "objectives" from milestone one will now be randomized objects throughout the map.

A pre-game menu screen will also be added which will serve as a more convenient way for users to enter server information with the addition of graphical options. After pressing Start on the menu screen, users will be taken to a lobby, where they will see the other users connected within the game.

Milestone 3 – Fog of War

Due by **March 20, 2014**. Fog of war will also be added as a feature in this milestone. Players on both sides will have a circle of vision, giving a live view of the players on the map. They will also share vision with teammates on the same floor.

The robbers will start the game with no vision, except for their own, and vision of their teammates. As they explore, they will reveal the map, example follows:

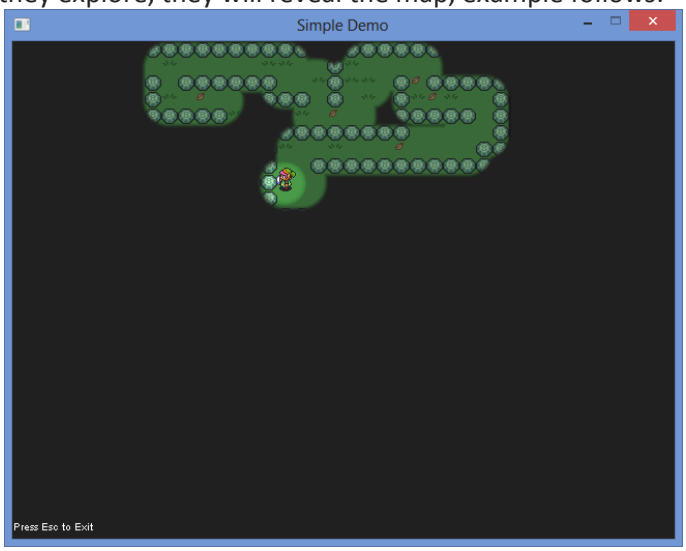


FIGURE 3: FOG OF WAR

The cops will start the game with vision of the map layout, as if they have explored the entire map, but not vision of opposing players. As example above, the cops view of the map is a darker shade where the map is revealed.

Team Members' Responsibilities:

Robin Hsieh

- Added objects
- Created multiple maps using the tiles system
- Implemented the stairs within the tiles of the maps
- Troubleshoot map spawning
- Troubleshoot collision system
- Added top floor detection dependent on the number of players

Sam Youssef

- Implemented and constructed the fog of war system
- Troubleshooted all fog of war issues
- Created cop spirits, and added player speech during gameplay
- Troubleshooted collision system
- Troubleshooted movement system

Tim Kim

- Added objects
- Created multiple maps using the tiles system
- Designed the parking lot as map
- Designed menu images
- Troubleshoot map spawning
- Troubleshoot collision system

Damien Sathanielle

- Added objects
- Created multiple maps using the tiles system
- Implemented the special tiles system
- Troubleshoot map spawning
- Troubleshoot collision system

Mateusz Siwoski

- Created all players
- Implemented the character animation system
- Player select screen, UI for the menu
- Added sound to whole game
- Implemented the Map system
- Implemented the Camera system
- Implemented the Render System
- Created the animation sequences (Intro/Title Screen/ Load screens)

Konstantin Boyarinov

- Created objects

Team Review:

As a team, our task was to implement the overall look of the game for all other components to use. Initially, the graphics system was the most important as having visuals was necessary for the other teams to begin proper testing of their features. After our system was implemented, our biggest goals were to take the time in created the actual look of the game. This would mean that we needed to create objects/animations and characters, and create different floors and walls. We used tools such as Photoshop, Illustrator, GIMP, Tiles, and a custom map editor to create all the tiles. These tiles grew from a start of 20x20 to eventually a set size of 40x40. Objects were constantly tested for various sizes, and eventually, with the correct implementation, we have been able to add objects of variable sizes as one complete object.

All the files the Graphics' group had impacted on, will be included in this CD.

Obstacles:

Of the tasks that we were assigned, our greatest obstacle as a group was content creation. We had great difficulty at the beginning, to create objects that were of the same look across all creators. It was only after having maps created along with objects, were we able to set a standard that we wanted for the look of the game. We used influences from classic Nintendo games to help design the various look of the characters, tiles and objects.