Application Note [s332100]

Single Player

The game can be played in sigle player board by selecting the `SINGLE BOARD` option in the main menu, using the joystick.

At this point a screen will appear that prompts the user to select the opponent, if `HUMAN` is selected a standard 1v1 game will start with turns alternating between the two players. If `NPC` is selected the user will be prompted to choose which color to play as, and then the game will start with the user playing against the computer.

Multiplayer

The game can be played in multiplayer mode by selecting the `TWO BOARDS` option in the main menu.

At this point a screen will appear that prompts the user to select which mode is he playing in, if `HUMAN` is selected the user will be playing standard against the other player, if `NPC` is selected the user will be replaced by the computer.

After selecting the mode the board will stop in a state of "waiting for connection", in this phase the board is waiting for an handshake to be exchanged with the other board (a move message with player id set to 0xFF), as soon as the handshake is successful we can proceed.

The next screen will prompt both player to choose which color to play as, the first to select will also decide the opponent's color.

At this point the game will start, each player can see the opponent's legal moves during the opponent's turn but cannot see the opponent's selector in real time to make the available time more meaningful.

Physical Connection

To connect the two LandTiger boards for the multiplayer `CAN1L` and `CAN1H` must be connected to the corresponding ones on the other board.

Controls

Players can move the selector for either walls or pawns using the joystick, clicking the joystick down will confirm the selection and change turn if the move is valid.

To place a wall the player must press `BUTTON1`, to change direction of the wall the player must press `BUTTON2`.

Implementation Notes

Displaying Images

To display images the GLCD library was used and extended, in particular a new font was added and the function `GUI_text` was renamed in `LCD_write_text` and extended to enable different font sizes. `LCD_SetPoint` was extended to support transaprent pixels, a number of accessory functions like `LCD_draw_image` where added. Sprites are defined in a `sprites.h` header file so that each can be esily drawn on screen by calling the `.draw` attribute:

```
typedef void (*Draw)(const uint16_t x, const uint16_t y);

struct Sprite
{
    const Draw draw;
    const uint16_t width;
    const uint16_t height;
};

extern const struct Sprite empty_square;
extern const struct Sprite player_white;
 ...
```

similar sprites are drawn starting from the same base images to save memory.

RIT usage

The RIT is used to both to keep track of the time and to handle the joystick and buttons. Handling of the latter is done in a way that eliminates bouncing and avoids moving by more than one step at a time.

CAN usage

The CAN peripheral is used to exchange moves between the two boards, ID 1 is always used, the move is passed in the 4 bytes of the `CAN_RxMsg.data` field.

NPC's decision making

I chose to categorize the NPC's moves in 7 distinct categories:

winning move a move that will win the game

blocking wall a wall that will block the opponent from winning

random wall randomly placed wall in the set of valid walls

random move randomly selected move in the set of valid moves

defensive wall a wall that prevents the opponent from blocking the npc's path, can be seen as a "safe path" for the npc, from the ordered list of candidate walls only the ones that won't make the npc's path longer are considered

offensive wall a wall that will make the opponent's path longer, again, from the ordered list of candidate walls only the ones that won't make the npc's path longer are considered

move the best move the npc can make by calculating the shortest path.

Every time a shortest path needs to be computed `A*` search is used, at the moment the weights are only modified to penalize moves that enable the opponent to jump over but there are the basis for implementing a tree of game states and using them to backpropagate the possible wins or losses through the weights.

The NPC's decision making is guided by chance, with hard coded probabilites for each category of move.