TECHNICAL DESCRIPTION SPRINT

Authors

Name	*Position*	*Email*
David Hedin	Operatons manager	davhe714@student.liu.se
Felicia Ringblom	Developer	felri885@student.liu.se
Jakob Boman	Quality Coordinator	jakbo921@student.liu.se
Viktor Stenström	Analyst	i11vikst@island.liu.se

Updates

Sprint	*Version*	*Date*	*Modification*	*Person in charge*	*Verified by*
3	1.0	2014-11-27	Describing main functionality	David Hedin	Jakob Boman
	1.1	2014-12-02	Adding new functions	David Hedin	Jakob Boman
	1.2	2014-12-03	Updated function description	Jakob Boman	David Hedin
	1.3	2014-12-09	Updated after inspection	David Hedin	Jakob Boman

Flow of calls in the code

This is a brief description of the flow of the calls in the code for developers to read before reading comments and code. This to give a general understanding of which function is called where and in what order. For detailed descriptions of all functions, either view the comments in the code or go to the github workspace for the application.

Game states

An important variable that is used throughout the game is the game_state variable. Possible values for the game_state variable is "Menu", "Game" and "Finished". It is used to determine which part of the game the application currently is running. Depending on the value of this variable, corresponding images will be shown on screen, only certain remote control buttons can be used described in **on_key()**, and certain functions will be active.

OnStart

This is the function that is called everytime the application is started. It will start by initializing variables for every menu setting that is later used to register the settings selected by the user for the game. The function will also determine which settings button that is highlighted when the menu is shown for the first time. Once everything is initialized, the **menu_start()** function is called to show the menu.

Menu

This is the main menu of the game. The functionality listed below runs once the menu_start() is called.

- game_state = "menu".
- **OnKey()** awaits user input. This input decides if settings or alternatives are to be changed and calls for the game to start.
- **change_setting()** decides which setting to change, by vertical navigation in the menu. For example, the user can here navigate to the menu field for selecting number of players.
- **change_alternative()** decides what alternative is to be used for the selected setting by horizontal navigation. For example, once the setting " number of players" is selected, the user chan here select the desired number of players.
- The UI for showing and changing the question categories does not behave in the same way as when changing the other setting alternatives. When changing which categories are activated the function **update_cat_choice()** is called.

Game

The game part of the application consists of three main phases: initiation phase, game phase and winning phase. Each of these phases are described below.

Initiation phase

When the game state is initiated a number of functions run to start the game logic and the graphical user interface of the game board. A majority of these functions use the user-selected menu settings as reference parameters to customize the game board to comply with the users wishes.

- When the game is initiated the global variable game_state is set to game_state = "game".
- The **init_players()** function uses the settings selected by the user in the menu to initiate a table of players where the number of players can range from two to six. Each player in the table is given an avatar and a point

counter.

- The **set_game_length()** function sets the length of the game by setting a limit to the number of questions needed to win a game. The game length depends on the menu setting chosen by the user and is saved in the global variable scores_to_win.
- **view.get_background()** is run to clear the screen from all menu elements.
- To display the player avatars on the game board during the first round the function **view.display_players()** is called.
- To display the goal line on the game board **view.display_goal()** is called. The goal line length depends on the number of players.
- To highlight the player whose turn it is **view.display_current_player()** is called. This will automatically be set to player number one during initiation.
- **get_question()** is called at the end of the initiation phase in order to load and display a question for the players to answer. The function calls upon **read_xml()** if there is no questions remaining in the game.
- **read_xml()** is used to load questions from an XML-file into the game.

Game phase

This is the main game loop where players answer questions and get points. When three (or the number of players minus one) has crossed the finish line, the winners are displayed.

- **OnKey()** waits until the all players has answered and then runs **execute_turn()**
- **execute_turn()** runs **change_current_player()** and then **display_current player()** indicate which players turn it is.
- **view.display_correct_answer()** runs to display which answer was correct and a placeholder for advertisement.
- **add_point()** runs to give points to the layers who answered correct and move the avatars accordingly on the game board.
- **remove_point()** is called upon if the game mode is “expert” or “go bananas” and the player is on a place indicated by a banana peel. The function then removes a point from the player if the answer is incorrect.
- **check_win()** controls if any player has received enough points to win the game. If three (or number of players minus one) player has received enough points, the game is finished, **view.display_winners()** runs to display the winners and the game state is set to “finished”. The game now enters the winning phase.
- If not enough players has received enough points to win yet, the game loop starts over again with a new call for **get_question()**.

Winning phase

When the winning screen is shown, the players are presented with two options, play again or quit to the menu.

- game_state = “Finished”
- **reset_game()** is called regardless of option the scores, winners and similar data is reset so that a new game can be started without any problems.
- **game()** is called if the play again option is chosen and a new game will be started in the same way as before.

Running on the box vs emulator

To run the code on the box, remove/comment the lines in the beginning of the main.lua file that are marked as emulator configurations.