User Documentation

The Yoyo application had been created with the aim to be easily usable by low-age kids. As a result, all possible actions are simple and evident, and the user is guided through the whole process of utilisation. This paper will still detail the list of possible interactions for each screen and the exact signification of visual information

Variable definition for the documentation
To facilitate scalability of this document, some variables will be defined here as they are values which could change during further updates

nbCharacters = 2
Number of playable characters

nbLevels = 5
Number of included levels

Main Screen
Definition:
The Main Screen is the first screen displayed. This screen is used as a home and to choose a character
Content:

- (2 + *nbCharacters*) elements without possible interactions (Wallpaper, Yoyo text at the top, and icon of each character)
- *nbCharacters* elements with possible interaction (Play buttons, each one will launch the game with the specific save and character noticed by the icon on top of it)
- During first launch, user will have to connect create a Database account. This is done automatically, but the user will have to select the Google Account he wants to link with database using Pop-up dialog box. For subsequent launches, the connection will be automatic

Level Selection Screen
Definition:
The Level Selection Screen is the one displayed after the Main Screen. Users can also return to it from each other screens (with the exception of the Main Screen) by pressing on system back button. It gives informations on current advancement, by changing state of icons. Users can use this screen to launch levels which are available.
Content:

- 2 elements without possible interaction (Wallpaper, Image of the character)
- *nbLevels* elements with possible interaction (Icon of the level, each which launch the level corresponding to the icon if this one is available)

Specificities

- Levels icon have 3 variation corresponding to the state of the level

    - Grayscale: The level is not available because the player didn't unlocked it yet. The interactions with these levels are blocked
    - With a flag: The level had been done and is available to be intended again
    - With a blue glowing: The level is the next, which means that it is not finished yet by the user and that he has to do it to unlock further ones.
- Character is always next to next level
- Displayed view is always centered on next level

Tree Level

Definition:

The tree level is one of the playable of the game. It aims to teach to associate numbers to figures. It is only accessible through *Level Selection Screen* and if the level is unlocked.

Content:

- 4 elements without possible interaction (Background, character, basket, figure)
- [1..9] elements with possible interaction (Berries)

Goal

The user has to drag each berry into the basket (in order to help youngest children, a simple touch on the berry will work). When all berries are dragged, level is considered as accomplished and the character will display an animation.

Developer Documentation

This Yoyo Project had been conducted using SCRUM lean development method. A specific had been taken concerning scalability for the code and it should be easy to add levels and characters. The following document aims to explain specificity and goal of each class and special functionalities to make subsequent developments easier.

Portrait Orientation

Done programmatically, each Activity has to call:

```
this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
```

Glide and Relative display

The project use an implementation of the Glide Project: https://github.com/bumptech/glide
This implementation can be called using an API defined in ***GlideApi***. It serves to programmatically display drawables more efficiently.

To make display relative, used implementation resids by 3 steps:

1) Catch screen size

```
int width, height, bck_width, bck_height;
// Get screen size
Display display = getWindowManager().getDefaultDisplay();
Point size = new Point();
display.getSize(size);
bck_height = size.y;
bck_width = size.x;
```

2) Select and set position by doing a fraction of screen size and using *ImageView* class

```
ImageView textYoyo = (ImageView) findViewById(R.id.main_screen_text);
textYoyo.setX((float) (bck_width * 0));
textYoyo.setY((float) (bck_height * 0.1));
```

3) Select wanted size by doing a fraction of screen size and set it using *Glide API*

```
GlideApp.with(this).load(R.drawable.yoyo).override(width,height).into(textYoyo);
```

Enumerations *Characters* and *Levels*

Contains respectively names of all selectionable Characters and implemented Levels

MainActivity* and Global Variable *selectedC

The MainActivity has for only goal (except being used as home) to set up global variable ***selectedC*** with the name of selected character. This variable is defined in ***GlobalApplication***, and it's the only use of this class.

Each class which needs the ***selectedC*** has to do create an object ***GlobalApplication***, initialize it using *getApplicationContext())* and then call *getGlobalVarValue()*

```
mApp = ((GlobalApplication)getApplicationContext());
selectedC = mApp.getGlobalVarValue();
```

Data Management

The main class managing datas is ***LocalData***. This class is called during several moment of the execution (In *OnCreate(), OnResume() and OnStop() of* ***LevelSelectionActivity*** and in the *onCreate()* of JULIO_CLASS_NAME)

Datas are under two forms:

- Local Data: The data stored locally are using the fact that class *LocalData* is serializable. An object of this class is used by *LevelSelectionActivity* and this object in then stored or load within memory. Here's key features of LocalData for local storage and loading:
    - *statut_levels* and one *statut_levels_charactername* for each character

This is object used to store the data. *statut_levels* is main one, and is loaded using *selectedC* global variable
    - *time*
      This variable is used to get current time while saving
    - *saveData(Activity pContext, Characters selectedC)*:
      The object passed by *pContext* will be stored in the form of bytes. The stored object can be found on user devices at the address: */storage/emulated/0/Android/data/com.example.jondhc.yoyo/files/yoyoSavedData.data*
    - *loadData(Activity pContext, Characters selectedC)*:
      Will try to fetch datas on device and put them inside the object passed by *pContext*
- Firebase: Saved datas are:
    - The character identified by his name
    - An int for each level (identified by his name) between -1 and 3. -1 is if level is next one, then ints represents number of stars obtained. If value is 0 then it means that level has never been completed
    - The current time of saving

Saving and loading logic:

- If no data is found then initial value is setted up in constructor LocalData()
- It's always intended to save data both locally and remotely. Value of the time is exactly equal for both among a same call.
- it's also always intended to load datas both locally and remotely. if a difference among time is found, most recent datas are preferred and the other one get aligned on more recent one (this case can happen if local file is deleted or if there is no internet during a save, leading to a failure of remote saving)

LevelSelectionActivity

This activity aims to display to the user which levels he did, which is next one and to make available ones launchables. Each level implements a pattern, and to add another level it's as easy a repeating the pattern, only changing values for position. This pattern start by the name of the level in comments and end by the call to ***GlideApi*** with the *ImageView* which will contain icon of the level.

Specificities

- Put grayscale on locked levels is done by *setLocked(ImageView v)*
- This class is in charge to load and save datas, therefore some calls to *saveData(...)* and *loadData(...)* are done in strategic moments
- Check for current personnage is not scalable and is currently done by a succession of *ifs* for each possibility
- Scroll view is centered on character by a simple call to *NestedScrollView.toScroll(int x, int y).* However and due to lifecycle, in *onCreate()* method this call has to be delayed, which is the point of *scrollToCharDelay()*