Project Proposal - Smiley Budding

Team: TYTS

Abstract

We will develop a virtual companion Smiley Budding that encourages users to smile more. Simley Budding uses the facial expression recognition technology to detect when a user is smiling when he/she is working on the computer. Budding will gain energy from users' smile and appear accordingly. We will use PyQT to build a GUI application and external facial recognition api to support the face recognition task.

The Team

Xi Lin (xl2793) - Full Stack Engineer
Ruolin Jia (rj2537) - Computer Vision Engineer
Lixing Zhang (lz2636) - Full Stack Engineer
Tianyang Yang (ty2388) - Database Engineer
Benhao Chen(bc2801) - Software Engineer

Background & Motivation

Smile is not only a responsive facial expression due to happiness, but happiness can be attributed to more smile. Research has found a strong link between regular smile and health and mood[1]. To encourage people to smile, we build the e-companion Smiley Budding that is 'fed' by users' smile. Budding is a popular emoji character among young people in China because of its lovely appearance. Budding has been downloaded

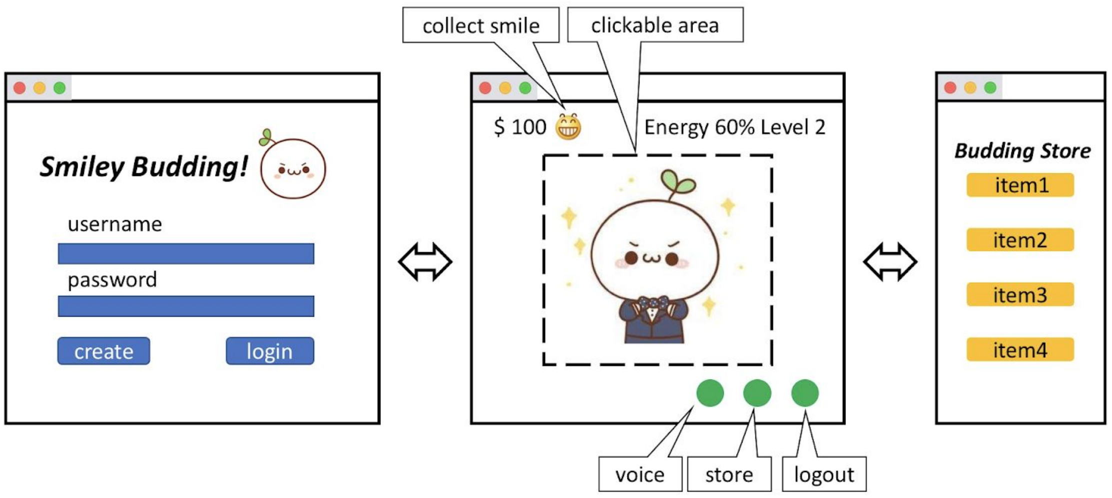
Software Requirement

Software Overview

The user can get a virtual companion (Budding). Once the software is running, Budding will appear on the screen. By clicking the "smile" button, the program starts to detect users' smile using a camera. The player can smile to the camera to earn virtual cash. Virtual cash can be used to buy food and items for Budding.
Budding will have some attributes such as Money, Energy, Level, etc. Items purchased from the store will increase Budding's energy. By patting the pet, the user can change the sticker of

[1] https://sunwarrior.com/healthhub/15-health-benefits-of-smiling

Pudding. Besides, Budding can upgrade and have more choices of stickers to show every time when the energy slot gets full.



User Story

We have one user type: the player who plays the game

(1) As a player, I want to create a new account of my own so that I can access to the system and check my private data.

Acceptance Criteria:

(a) The player can create a new account/password in database

(b) The player can login with correct account/password information

(i) If the password is not correct, reject attempt to login.

(c) Retrieve the data of the pet using the account-specific information

(2) As a player, I want to smile to earn virtual cash so that I can gain virtual money

Acceptance Criteria:

(a) When the user clicks “making money” button, the program can detect user’s smile within 3 seconds

(b) After detecting a smile, the player will see his/her money is increased.

(3) As a player, I want to purchase food from a store (showing a list of items), so that I can feed Budding with food.

Acceptance Criteria:

(a) The player can spend money to buy items from the store

(i) If user does not have enough, money, the purchase should be denied.

(b) After the player bought the item, Budding will consume the item directly

(c) Items from the store will increase Budding’s energy.

(d) Once the energy reaches an upper bound, Budding can level up.

(4) As a player, I want to pat Budding, so Budding can change the appearance

Acceptance Criteria:

(a) Budding is clickable.

(b) When the player clicks Budding, the picture of budding will change the appearance

(5) As a player, I want to click the music button, so that I can hear budding's voice.

Acceptance Criteria:

(a) When the user clicks the music button, if currently no song is being played, a new random song will be played. Else, it will stop playing the current song.

Tentative user stories (Wishlist):
if more testing needs to be done in this course, we could add functions to our implementation from this wishlist

1. As a player, I want to send Budding for an adventure, so he can make some virtual cash from it (The interface can be a little gamble game: you flip a card and it will bring you good news or bad news during the adventure)
2. As a player, I want to control Budding and let it jump or move around so that I can gain a better experience of interaction with it
3. As a player, I want to buy houses for Budding, so it can live in a new environment
4. As a player, I want to access my smile data by plots so that I can know the total scores and compare with others

Technical Summary

Language and Platform

Python3, Mac

External API

Microsoft Emotion API - facial expression recognition

Backend

PyQT5 - GUI backend
SQLite - Data storage
Matplotlib - Data visualization

OpenCV - face capturing

Frontend

PyQT5 - GUI frontend
QT Designer - GUI frontend designing

Other Tools

Pylint - Code static analysis
Github - Version control
Budding Pop emoji - Graphics

Project Risk & Contingency Plan

Why this project is doable?

The Project utilizes various open-source tools: we use Qt designer to develop the user interface, PyQt5 to react to user's interaction events (e.g., clicking), external API to recognize people's smile, SQLite to store users' accounts information, and Github to realize version control. Pre-trained facial expression machine vision model is also available from various online sources. Therefore, this project is technically manageable.
The idea ,requirement and management of this project is neither too difficult nor trivial so the project is highly doable and meaningful.

Potential Risk & Contingency Plan

Risk 1:
The open source pretrained detection algorithm may have an unaccepted recognition accuracy and the software cannot detect smiles of user. We also don't have enough dataset and time if we want to build our own machine learning model.

Contingency Plan 1:
In our demo-version, we replace the smile detection module with a clickable button, the user gets scores and Budding gains energy when the user clicks the button, or we only use face

detection rather than smile detection for demo purpose. Face detection is much more tractable than smile detection.

Risk 2:
We designed three GUIs for the player (login, main and item store), we planned to use three different windows. It brings extra works of building the multi-interface application and we may not have enough time to finish the whole project.

Contingency Plan 2:
We can add all three modules into one single GUI, in this case we don’t need to spend time on GUI switching.

Risk 3:
The project is not complicated enough to fulfill the testing requirement in the course.

Contingency Plan 3:
We can always add new functions if we think the testing scope is not enough. We will list some tentative features of the project so we can implement them later if needed.