Multimodal Interfaces

chickenclimb iOS Game

Felipe Kaufmann

University of Fribourg

Fribourg, May 27, 2011

Contents

1 Overview & Game Idea **1**

2 Architecture **3**
2.1 Hardware 3
2.2 Software 3
2.2.1 Apple iOS SDK 3
2.2.2 Cocos2D for iPhone 3

3 Implementation **4**
3.0.3 Main Event Loop 4
3.0.4 Input Handling 4
3.0.5 Tracking of Chicken Movement 5

4 Modalities **7**
4.1 Used Modalities 7
4.2 CASE &CARE Models 7
4.2.1 CASE 7
4.2.2 CARE 8

5 User Evaluation **9**

6 Conclusion **11**
6.1 Further Improvements 11

7 Resources **12**

Chapter 1

Overview & Game Idea

The basic idea of the game is to climb as high as possible with the chicken, jumping from platform to platform, and while climbing up, to collect as many coins as possible. In order to jump the user can either use his voice or the touch screen. The lateral movement of the chicken is controlled by the built in iPhone accelerometer.

The initial project idea consisted in a collection of a few very simple iOS games and challenges which would be played in turn by multiple participants, the games appearing in random order but increasing in difficulty, until all player would be disqualified. Due to time constraints and issues with the voice recognition library that would handle the switching of players, the project has been de-scoped. Instead of working on multiple games, only one of the initial games has been picked out to be further developed, resulting in the chickenclimb game.

Figure 1.1: Application Screenshot

Chapter 2

Architecture

2.1 Hardware

An Apple iPhone 4 and an optional Bluethooth Headset are required to run the game. In order to build the game, a system running Mac OSX and the latest iOS SDK are required.

2.2 Software

2.2.1 Apple iOS SDK

As any other iOS Application, the game uses the *Foundation Framework* and the *UIKit* library in order to integrate with iOS. Additionally, the *AV-Foundation* Framework was utilized to record sound input.

2.2.2 Cocos2D for iPhone

Most part of the game is based on elements of the *Cocos2D for iPhone* game framework, which is open source and based on the Cocos2D python framework.

Chapter 3

Implementation

3.0.3 Main Event Loop

The essential game mechanics happen within the step method of the Game class, which is called repeatedly and represents the main game loop:

- Handle the orientation and scale of the chicken
- check and prevent chicken from getting out of bound
- set bird velocity and position
- check for sound input
- distinguish if chicken is falling or rising and handle accordingly

3.0.4 Input Handling

As discussed, the handling of sound input happens within the main game loop: The recorder which is constantly recording sound is checked for the peak volume, and, if a certain threshold is surpassed, the jump method is called.

The touch and gesture input however, are of evented nature: Hooks provided either by the Layer class of Cocos2D (touch: ccTouchesEnded) or

```
/*
 * Handle sound input
 */
[recorder updateMeters];
audioPeak = [recorder peakPowerForChannel:0];
if(audioPeak > audioThreshold && [self isJumpAllowed]) {
    [self jump];
}
```

Figure 3.1: Sound Handling

UIKit (accelerometer, UIAccelerometer#didAccelerate) are triggered once interaction occurs.

3.0.5 Tracking of Chicken Movement

The movement of the chicken are mainly determined by assigning and tracking variables for position, velocity and acceleration. Furthermore, the chickenState variable tracks wether the chicken is falling, rising, or standing still. The acceleration is used as a negative or positive multiplier in order to gradually make the chicken rise or fall.

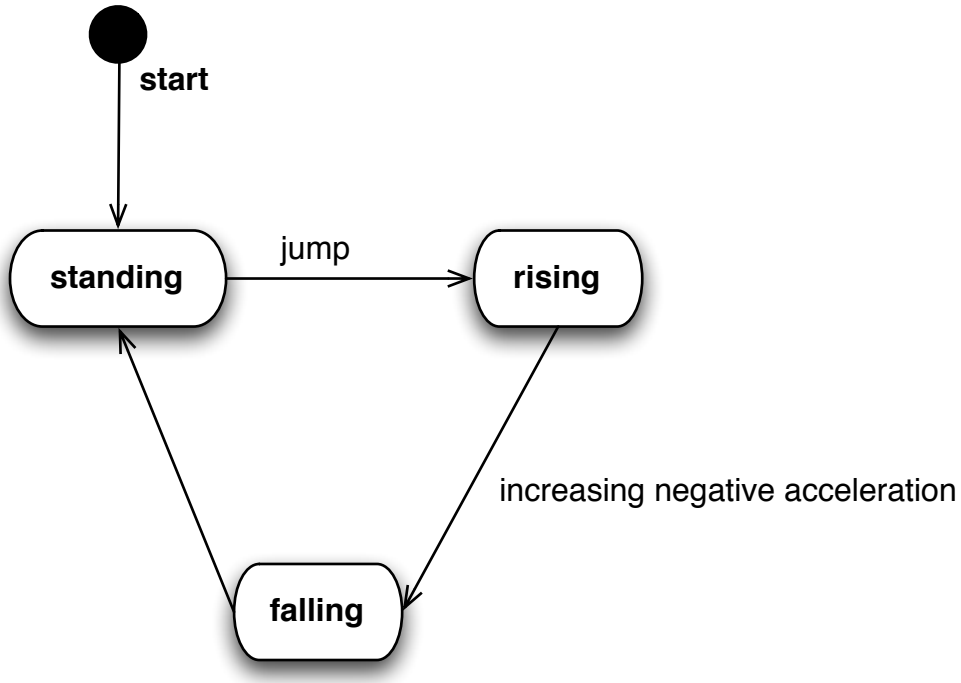

Figure 3.2: State Diagram

Chapter 4

Modalities

4.1 Used Modalities

The input methods provided in the game are:

- Speech, captured trough the microphone, in order to jump
- Touch, captured by the touchscreen, in order to jump
- Gesture, captured by the accelerometer, in order to control direction

The user is free to choose wether he uses speech or touch interaction in order to make the chicken jump.

4.2 CASE &CARE Models

4.2.1 CASE

Two cases can be distinguished in order to classify the communication type of the modalities: As the user can either use speech or touch interaction to trigger a jump, the use of these modalities are *exclusive*, as no two jumps can be triggered at the same time. The use of touch or speech together with the accelerometer to determine direction are of *alternate* nature however: First

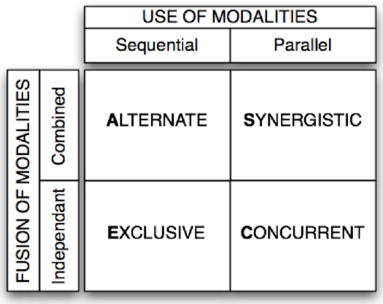

Figure 4.1: CASE Model

a jump is triggered, using either voice or touch, then the accelerometer is used to determine the jump direction. One would be tempted to classify this as a synnergistic relationship, however, triggering and then steering happen in sequence and not in parallel.

4.2.2 CARE

In terms of usability properties, again, the two cases can be distinguished: The choice between touch or voice for jumping is of *equivalent* nature, as only one modality has to be used and no parallelism or sequence are presented. The combination of touch or voice together with the accelerometer however is *complementary*: Two of the three modalities have to be used in sequence, and no modality alone is enough to advance the game.

Chapter 5

User Evaluation

The method of User evaluation used for this project was the constructive interaction method, which is of qualitative nature. Though no measurable output could be gained by this, the results of the user evaluation lead to easy solutions to significantly improve the usability of the game. Besides pure usability tweaks, User Evaluation was used throughout the whole development process in order to produce a playable game, as game mechanics and input handling had to be adjusted and fine-tuned to produce an enjoyable game. Some findings were:

- The adequate level of sound input to trigger a jump was determined by user evaluation.
- Movement speed and the rate at which the chicken can jump were adjusted based on user evaluation.
- Lateral movement of the chicken is disabled once horizontal movement ceases: this prevents the bird from sliding from the platforms when standing, which was of great annoyance to the test candidates.
- The backdrop color was adjusted to offer better contrast with the rest of the game elements, as some users with sight impediments had

difficulty to track the movement of the chicken.

- The user is not instructed that he can touch or speak to make the chicken jump. An introductory screen or a tooltip at the beginning of the game would solve the problem. This has not yet been implemented.

- Some users held the device very still, or positioned it on a flat surface. Missing any significant accelerometer input, the bird would not move sideways, making it impossible to reach some platforms. Increasing sensitivity has partially mitigated the issue. Better results could be achieved by instructing the user on screen about the accelerometer usage.

- When triggered by audio, it could happen that the chicken jumps twice when it touches a platform on the way up (as seen in demo video, fixed by now)

Chapter 6

Conclusion

Lacking any experience in game development, the project was quite challenging and had great learning effect. Especially the fine tuning of game mechanics as movement speed or collision detection were totally new concepts, and could still be improved. On the other hand, the usage of multiple modalities was much easier as initially foreseen. Working with the Cocos2D framework showed me that while working with standard iOS view elements is quite easy and intuitive, adding custom graphic elements and layering adds a whole new level of complexity. The lack of experience also lead to slightly unorganized code that invites refactoring, and which would look significantly different on a second attempt, with the knowledge gained so far.

6.1 Further Improvements

The two main issues with the current game that invite further work are the missing instructions to the user, mentioned in the User Evaluation chapter and a few collision detection bugs as the chicken does not always stand exactly on the platform, but also ”inside” it or hanging on it.

Chapter 7

Resources

- Apple iOS SDK documentation: http://developer.apple.com/
- http://www.cocos2d-iphone.org/wiki/doku.php/
- http://www.mobileorchard.com/
- http://stackoverflow.com/questions/tagged/objective-c