Sensei App
Product Design Document

Author: Atacan Demiralp

CONTENTS

GENERAL DESIGN

Product Overview

Title: Sensei App
Product Type: Mobile Application
Platform: IOS
Genre: Augmented Reality
Target: Anime fans (aging from 12 - 30)
Release date: 16 February 2018
Current situation: Development build
Published: No

Sensei App is an augmented reality app which allows user to interact with digital sensei in real world. User places the sensei in a place and asks him to do the certain activities either by clicking interface buttons or recording speech. It will be a development build which is developed to test Apple ARKit development kit and speech recognition platform Wit.ai of Facebook. The secondary goal is to make user have fun by letting user to see in real life something that belongs to the anime Naruto Shippuuden.

Concept

The sensei Kakashi is the main and only character in the application. He has something to say and some skills, called jutsu to show. User who is an anime fan probably knows this character and his skills. First, user asks Kakashi to show some jutsu that are defined, then he replies and take some action.

Unique Points

- Markerless augmented reality experience
- Interaction with speech
- Trained custom intents on Wit.ai platform
- Unique animations and impressing particle effects
- The exact sounds that belong to the anime

Minimum Requirements

- Unity3D v5.6.2 or higher to develop the application
- Blender3D version 2.79 to graphic and animation design
- Apple ARKit 1.0 Framework to use augmented reality
- Xcode 9.x with latest IOS SDK that contains ARKit Framework to get build for IOS
- IOS device that supports ARKit (iPhone 6s or later, iPad 2017 or later) to run the app
- IOS 11 or higher version operating system to run the app

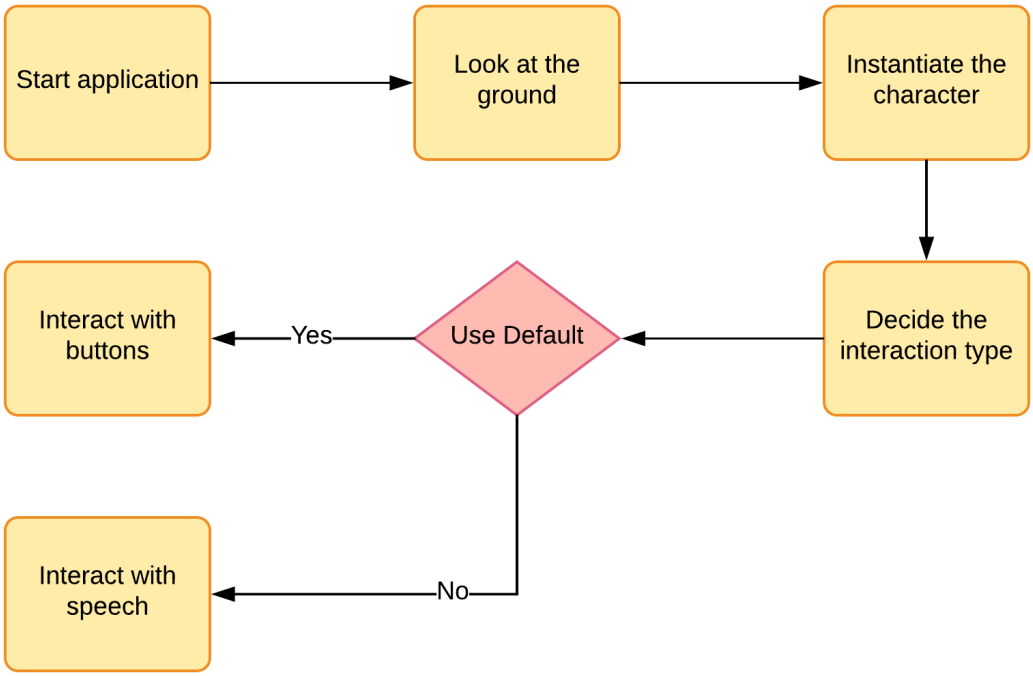
Application Objective

The objective of the application is to ask sensei Kakashi to do something and watch him doing in real world.

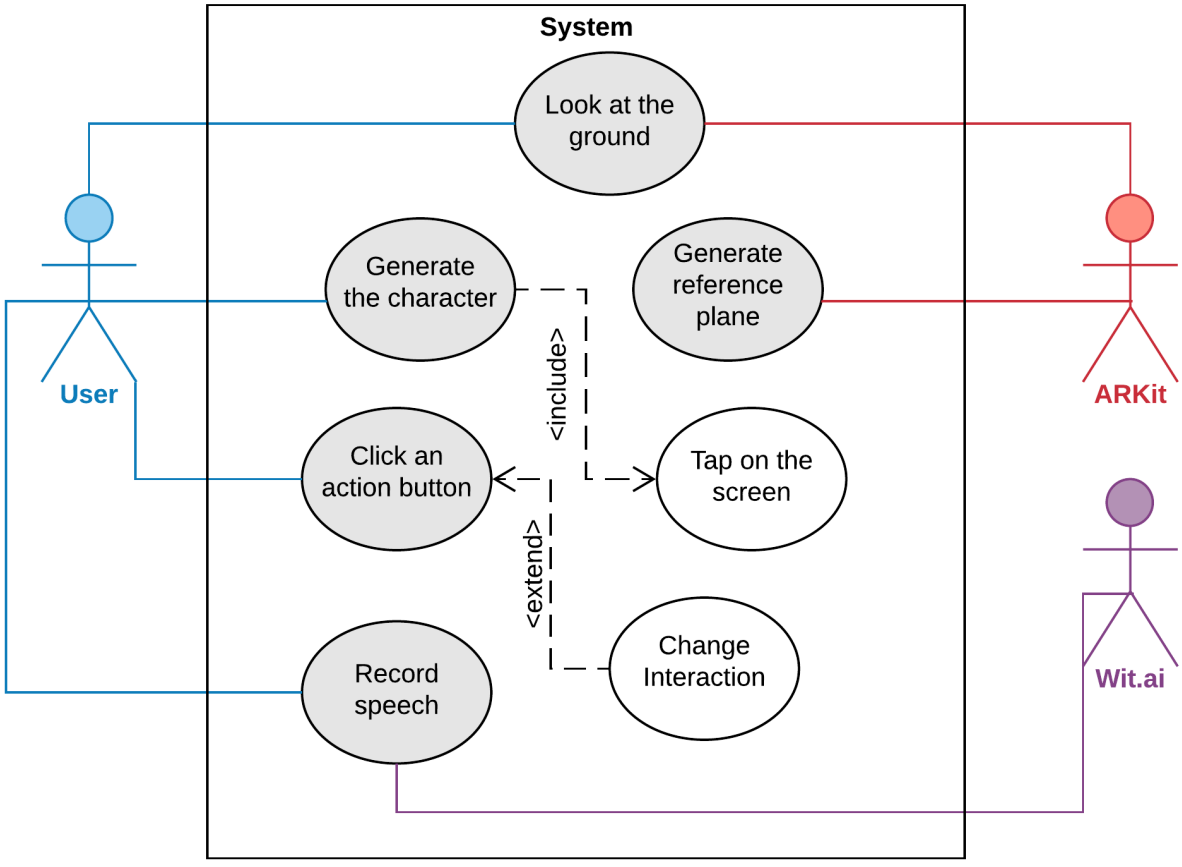
Non-Functional Requirements

- User should experience augmented reality with no marker images.
- User should see a reference displaying whether the camera recognizes the ground
- User should tap on the screen of iPhone to generate the main character.
- Tap point on the screen should be projected to the ground and defined as the location point of the main character.
- User should change interaction type by clicking only one button.
- Buttons should tell what they do.
- First, user should click a button, then the related animation and voice should be played.
- Animations should look natural as possible.
- Voices should be clean and exactly the same as those in the anime.
- User should record speech to command instead of using interface buttons.
- Speech commands should be simple.

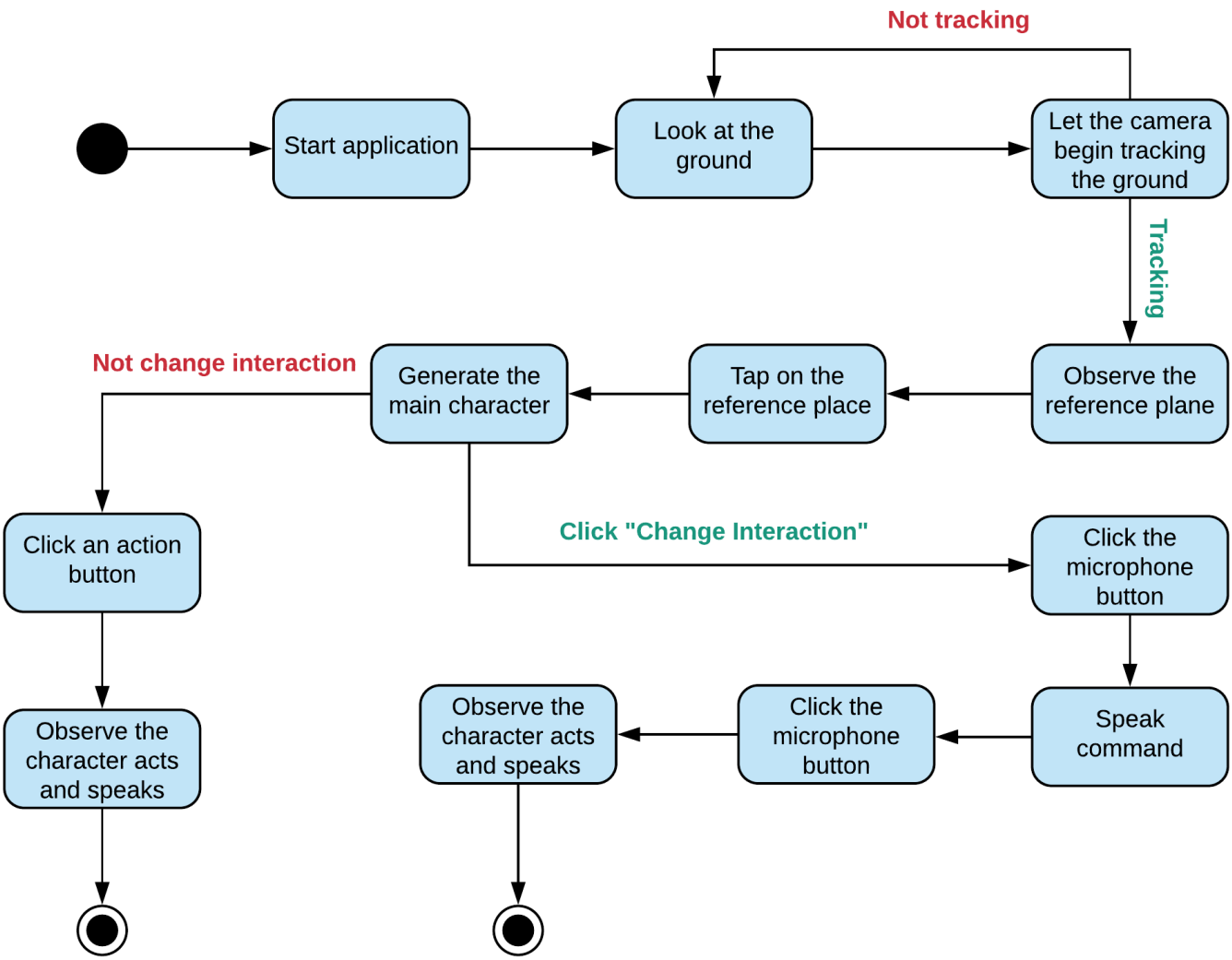
Basic User Flow



Basic Use Case



User Journey

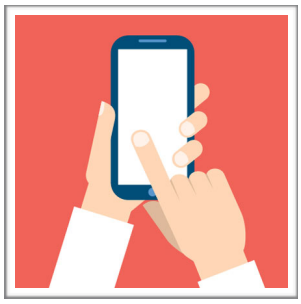


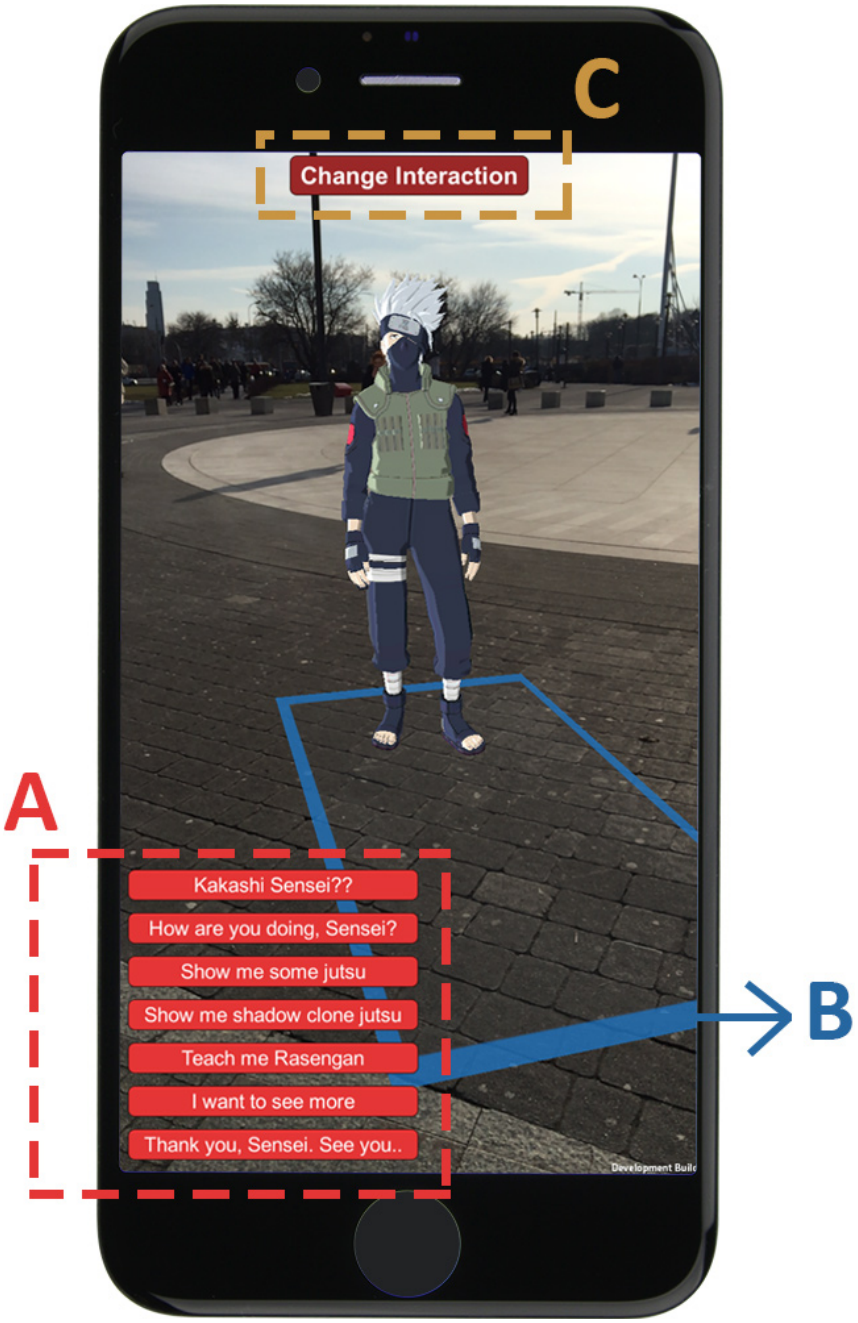
Game Play

Controls

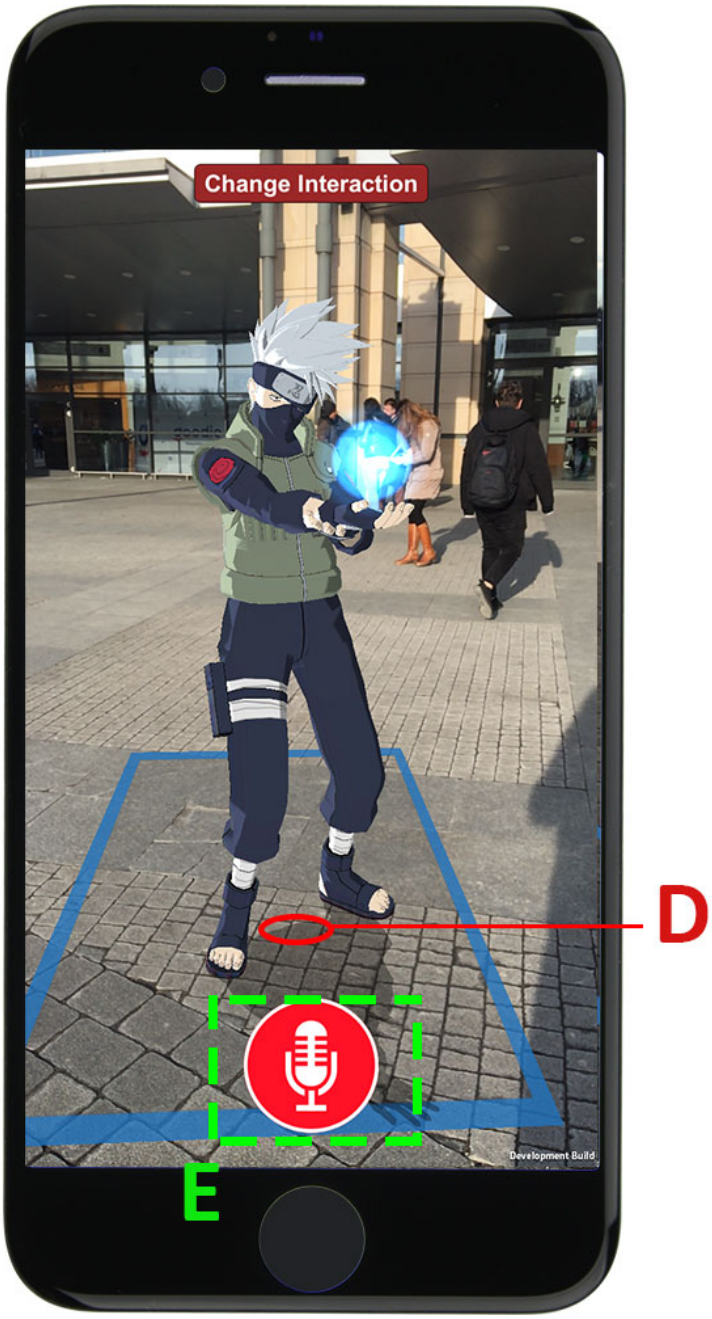
With touch control, user first taps on the screen. The tapped position on the screen is converted to the position on the reference image on the ground. Second, user clicks on the buttons by tapping. In the default interface, there are six action buttons, each one calls a different function, and one button to change the interaction type into speech mode. When user switches the interaction type, the microphone button appears. When it is clicked, it begins to record voice and it keeps recording until it is deactivated by clicking it again.

Touch Control:





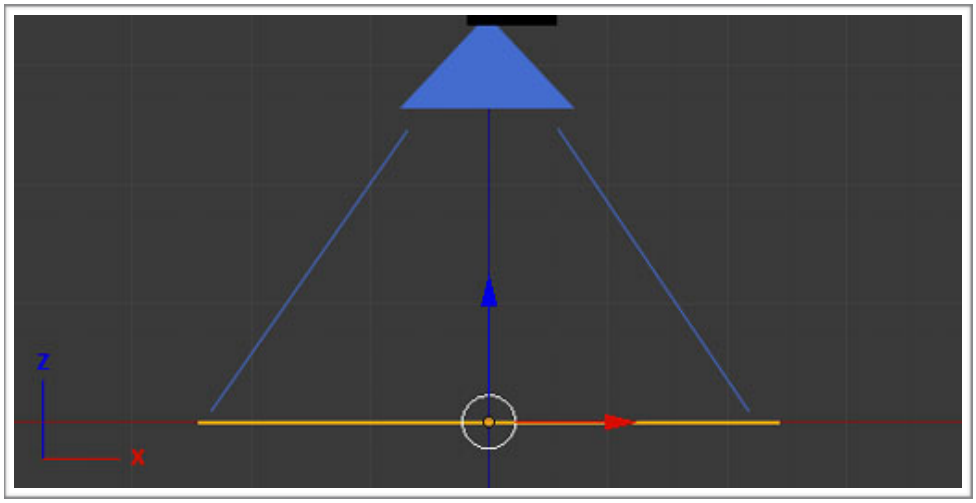
A: Action Buttons
B: Reference Plane
C: Change Interaction Button



D: Touch point
E: Record button

Camera

The AR* camera is same as the normal phone camera. Once the application starts, the AR camera opens automatically and begins to search for flat ground to recognize and track. The camera should be horizontal to the ground to make the AR camera recognize the ground. If the camera points forward, upward or any direction but downward, tracking process does not begin. After the camera recognizes the flat ground, it begins to track and draws blue rectangle to let the user know that the are inside the rectangle is being tracked.
(*Augmented Reality)



The Character

Main Character

Kakashi Sensei (sensei means teacher in Japanese) is a teacher in the ninja academy in the anime Naruto (which originally belongs to the manga Naruto, written by Masashi Kishimoto). He is knows as “The copy ninja” because he has the ability of copying ninjutsu skills of the opponents. He has more than thousand of skill, called “jutsu”, and he is master of almost all of them. He is extremely skilled, humble and patient. He is one of the most beloved character in the manga/anime.

Character States

Idle: The idle state is cycled animation where Kakashi Sensei is staring at the camera with empty looking and relax.

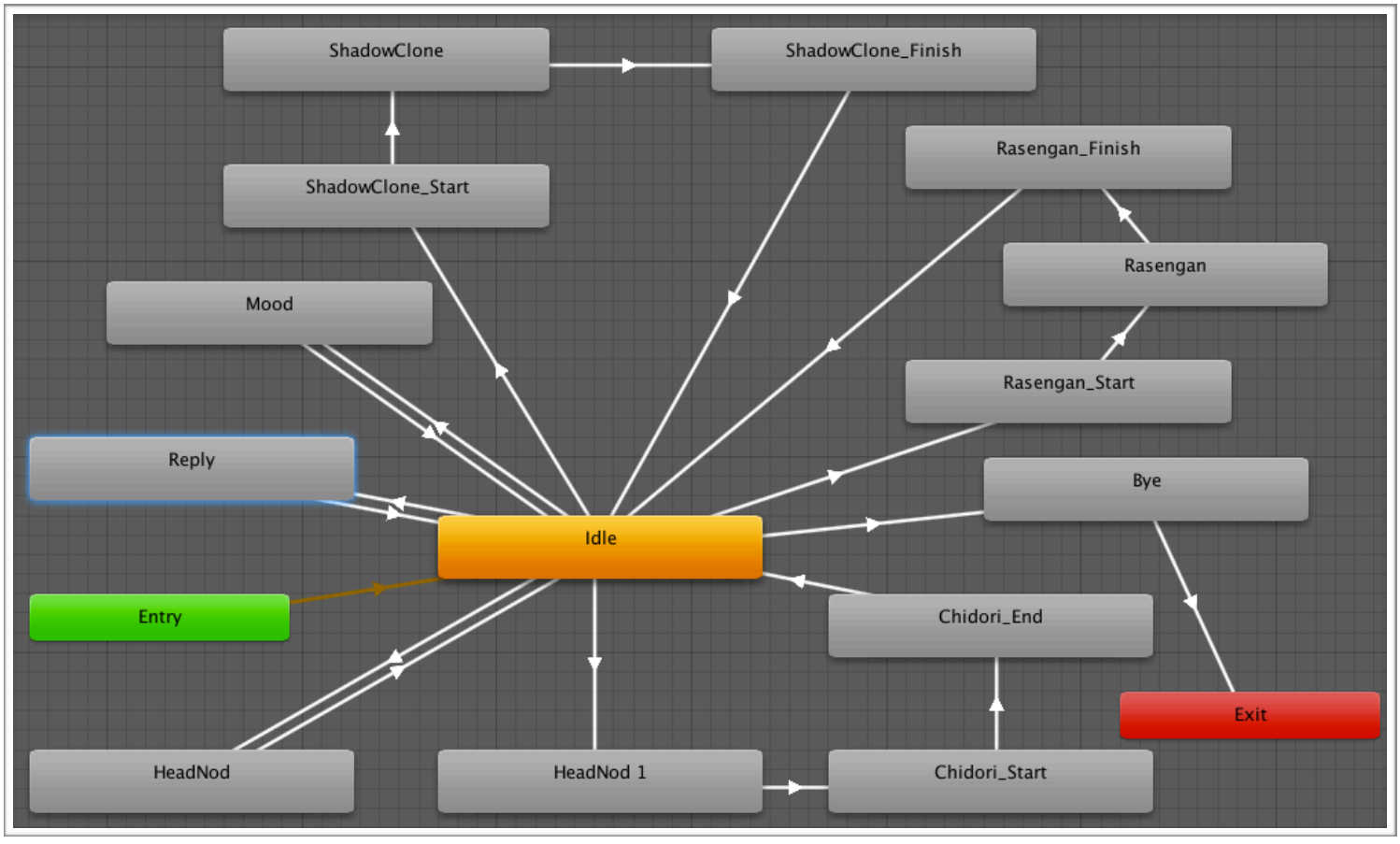
Reply: The state where the idle animation plays with a sound clip, saying “Yes”.

Mood: The state where Kakashi Sensei is showing some gestures.

HeadNod: The state where Kakashi Sensei is nodding and saying he is busy.

ShadowClone_Start: The state where Kakashi Sensei is starting to do Shadow Clone Jutsu.
ShadowClone: The state where the shadow clones are being generated.
ShadowClone_Finish: The state where the shadow clones are vanishing.
Rasengan_Start: The state where Kakashi Sensei is starting to do Rasengan.
Rasengan: The state where Kakashi Sensei is keeping his hands still, holding chakra ball.
Rasengan_Finish: The state where Kakashi Sensei is quitting holding the chakra ball.
HeadNode 1: The state where Kakashi Sensei is nodding before showing Chidori.
Chidori_Start: The state where Kakashi Sensei is doing hand seals and generating lightning in his hand.
Chidori_End: The state where Kakashi Sensei is quitting holding the lightning.
Bye: The state where Kakashi is thanking you for coming, then disappearing.

Character State Machine

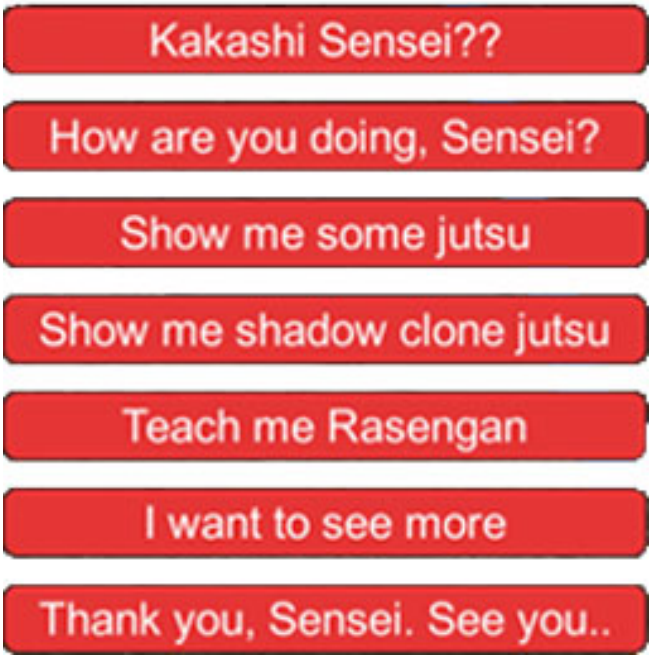


Art

UI Elements

Button to switch interaction

Record button



Action buttons

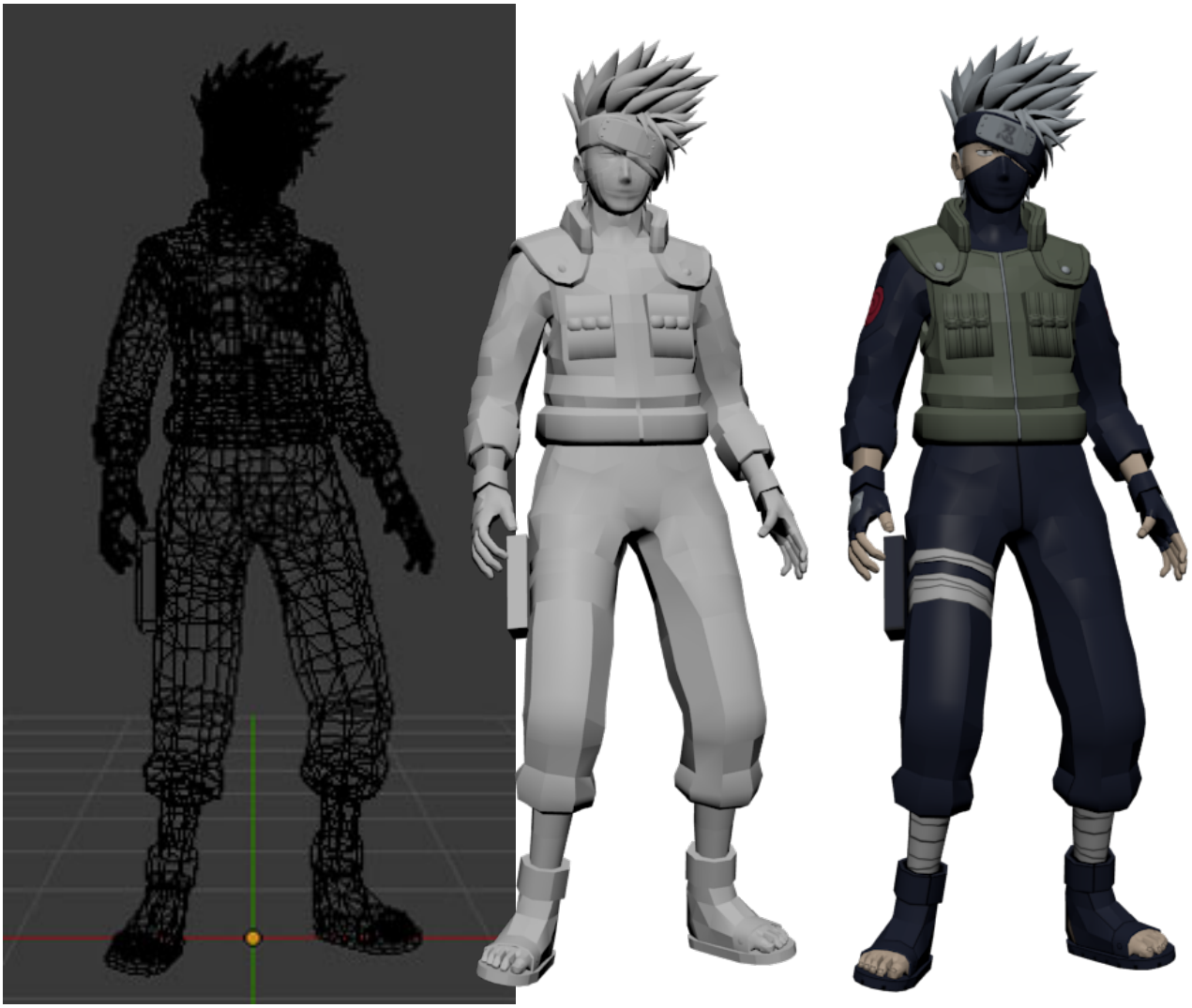
Record button during recording

App icon

UI Color Palette

3D Graphics

Kakashi Sensei:



Vertices: 11,651
Faces: 11,060
Trigons: 11,060
File type: .blend

Animations

NAME	FRAME COUNT
Idle	58
Mood	93
HeadNod	95
ShadowClone	99
Rasengan	140
Chidori	81

Sound Clips

NAME	CATEGORY	RUNS IN STATE
yeah.mp3	Character speech	Reply
Icantchat.mp3	Character speech	Mood
Imbusy.mp3	Character speech	HeadNod
letsbegin.wav	Character speech	ShadowClone_Start
bunshin.mp3	Special effect	ShadowClone_Start
instantiate.mp3	Special effect	ShadowClone
destroy.mp3	Special effect	ShadowClone_Finish
understood.mp3	Character speech	Rasengan_Start
rasenganFX.mp3	Special effect	Rasengan
gotit.wav	Character speech	Rasengan
stillwannago.wav	Character speech	HeadNod 1
seal.mp3	Special effect	Chidori_Start
seal1.mp3	Special effect	Chidori_Start
seal2.mp3	Special effect	Chidori_Start
seal3.mp3	Special effect	Chidori_Start
chidori.mp3	Special effect	Chidori_Start
thanksforcoming.wav	Character speech	Bye

TECHNICAL DESIGN

Technical Summary

Sensei App will be developed in approximately one week by me using the Unity3D game engine. For 2D assets, Adobe Photoshop will be used. For 3D asset, the graphic model of Kakashi will be searched and found in the Internet. It will be edited and rigged in Blender. The animations will also be created in Blender. The sound clips will be searched and found in Internet, also will be recorded, then cleaned and edited in Audacity. Particle effects will be created in Unity3D game engine. For speech recognition and machine learning, Facebook's Wit.ai platform will be used.

The application will be deployed for IOS only.

Equipment

Hardware

I will be the only member in the team. I have 13" MacBook Pro (late 2011) which causes very slow building in both Unity and Xcode. As the test device, I will be using my own IPhone 6s.

PRODUCT	TASK	COST	QUANTITY	TOTAL
MacBook Pro 13"	Asset Creation, Game Development	1300 USD	1	1300 USD
IPhone 6s	Test Device	450 USD	1	450 USD
				1750 USD

* It is assumed that I do not have the hardware.
** Assumptions made according to current price updates in the project time

Software

All the software used for the development of Sensei App are listed below.

PRODUCT	TASK	COST	QUANTITY	TOTAL
Unity Personal	Game Engine	0	1	0
Blender 3D	3D Asset Modeling, Animation	0	1	0

PRODUCT	TASK	COST	QUANTITY	TOTAL
Adobe Photohosp	2D Asset Creation	120 USD	1	120 USD
Audacity	Sound Editor	0	1	0
XCode	IDE	0	1	0
ARKit	AR SDK	0	1	0
Wit.ai	Machine Learning Platform	0	1	0
Google Office	Project Management	0	1	0
				120 USD

* It is assumed that I do not have the software.
** Assumptions made according to current price updates in the project time

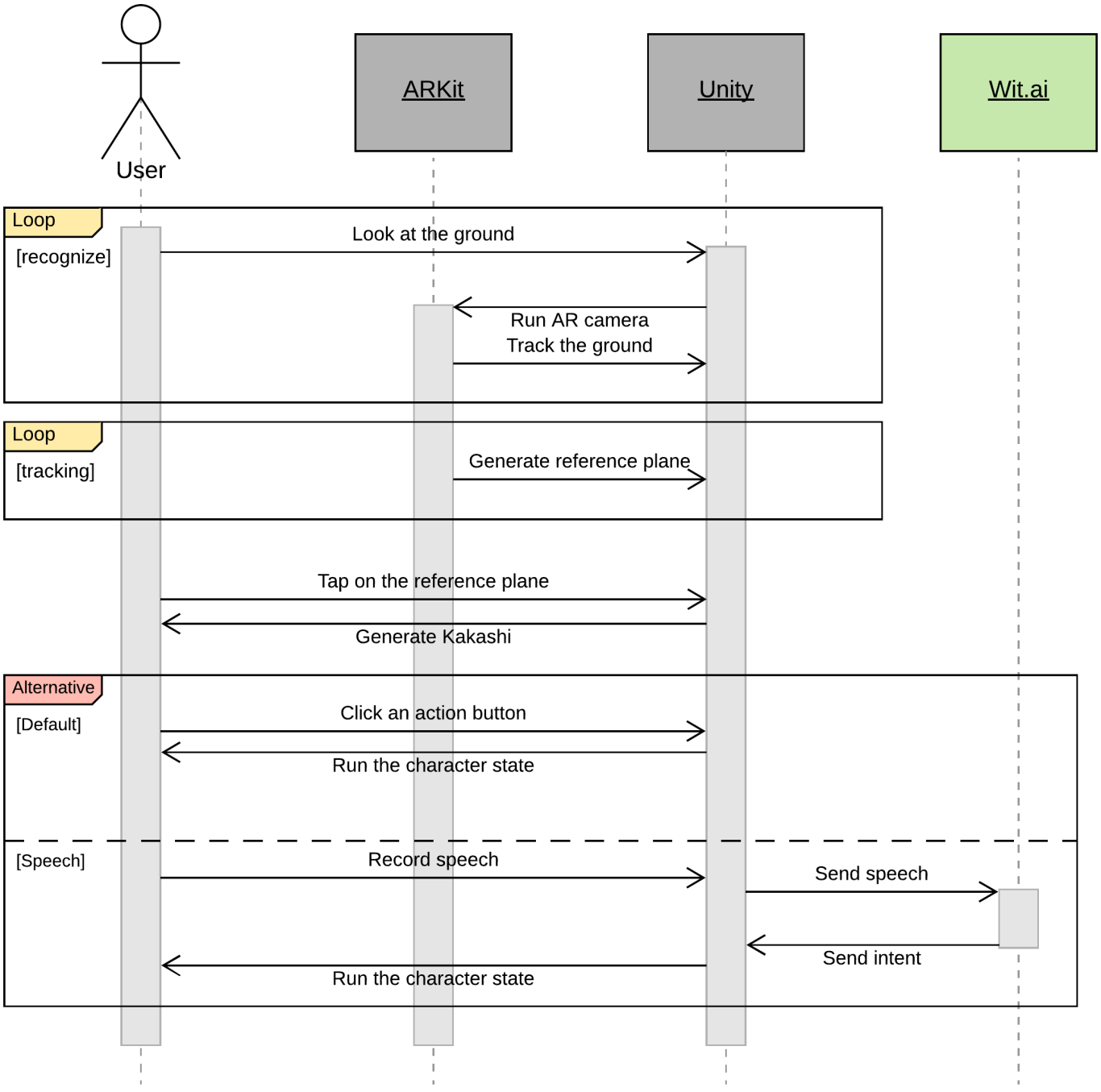
Scheduling

Development Plan

	Research	2D Art	3D Assets	Scripting	Audio	Test
Day 1	Search for the 3D model		Edit the 3D model			
Day 2	Search for animations		Create animations	Begin ARKit		Test ARKit
***Day 3**	Search for sounds			Character states, animations		Test animations
***Day 4**	Learn Wit.ai	Create UI		Character states, Train Intents	Edit sounds, record sounds	Test animations and intents
***Day 5**			Create particle effects	Particle effects, train intents, add sounds		Test effects, sounds and trained intents
***Day 6**				Voice recording, JSON parse		Test UI and speech recognition
***Day 7**				Fine tuning, deploying to IOS		Overall Test

*** Milestones**

Sequence Diagram



Training The Intents

To use machine learning technology of Wit.ai, sound clips need to be recorded and sent to Wit.ai, then Wit.ai platform needs to match the clips to the specific intents. Each intent returns with a confidence level showing that how accurate the given sound fits an intent.

Intents	Speech Commands
sensei	Sensei
sensei	Kakashi Sensei
howru	How are you
howru	How are you doing
howru	How are you Sensei
imbusy	Show me some jutsu
imbusy	Show me your arsenal
lets_begin	Show me shadow clone jutsu
lets_begin	Shadow clone jutsu
teach_me	Teach me rasengan
teach_me	Teach me
teach_me	Rasengan
more	I want to see more
more	I want to see one more jutsu
more	Chidori
thanks	*No need to highlight a specific part in the sentence

Some speech records accurately matches to the intent. In the example, the command “I want to see more” returns the intent “more” with a confidence level 0.997.

However, some commands are not understood by the platform because the command contains foreign non-English words, such as “Kakashi Sensei”. The learning platform understands “Kakashi” as “could crash she”, “cockroaches she”, “could crash is to see” or similar meaningless sentences. In this case, these meaning sentences also need to be validated to match the word “Kakashi” , so that even if user commands “Kakashi Sensei” and the system understands “could crash is to see”, then the system returns “sensei” intent which will be triggering the reply function of the character.