Tiger eyes

Technical Manual

Team name: FP
Team members: Chong Sun, Bo Li
Email: *chongs@clemson.edu*

Session 0: Overview

Tiger eyes is an augmented reality application of IOS. Tiger Eyes will operate on an iPhone with iOS 7 or later. Connectivity from a wireless carrier or a Wi-Fi provider is required to make Tiger Eyes work. And we highly suggest you to turn on the Wi-Fi to improve the accuracy of the location.

Session 1: Background

Augmented Reality (AR) was first demonstrated in the 1960 recently have technologies emerged that can be used to easily deploy AR applications to many users.
Camera-equipped cell phones with significant processing power and graphics abilities provide an inexpensive and versatile platform for AR applications.

Session 2: Augmented Reality

Augmented reality (AR) is a live, direct or indirect, view of a physical, real-world environment whose elements are augmented (or supplemented) by computer-generated sensory input such as sound, video, graphics or GPS data. It is related to a more general concept called mediated reality ,in which a view of reality is modified (possibly even diminished rather than augmented) by a computer. As a result, the technology functions by enhancing one's current perception of reality. By contrast, virtual reality replaces the real world with a simulated one. Augmentation is conventionally in real-time and in semantic context with environmental elements, such as sports scores on TV during a match. With the help of advanced AR technology (e.g. adding computer vision and object recognition) the

information about the surrounding real world of the user becomes interactive and digitally manipulable. Artificial information about the environment and its objects can be overlaid on the real world.
40 Best Augmented Reality iPhone Apps

Session 3: About Tiger eyes

With Tiger eyes, users can see more and get more information which can't be seen from their eyes. The application mainly implements three functions:
1.Get building information (including the names of the buildings and the distance from the buildings) through camera.
2.Record the place of your car and find it.
3.Decode QR code and get more information.
To implement these functions, we mainly used following technology:

Augmented Reality technology is key of our application. The main functions of the application are all based on augmented reality. We need to use AV Foundation framework to accomplish the functions, which provides an Objective-C interface for managing and playing audio-visual media in your Mac OS X application. Thanks to AR kit, an open source augmented reality library, which help us to implement the augment reality effect easily.
Reference: *AvFoundation.framework, AR kit.*

Session 4: Location

The Core Location framework lets you determine the current location or heading associated with a device. We use Google's Places API ,a service that returns the information of the places, that loads a list of POIs near user current location.
Reference: *CoreLocation.framework, Google Place API.*

Session 5: QR code decode.

To decode QR code, camera must be used. So we need AvFoundation framework.

We used AVMetadataMachineReadableCodeObject (it is a cool new feature in iOS 7) to decode the QR code.

Then a highly customized alert box pop up. With Richart's open source code we are able to use it.

We also use remoteimage view from Adrian Geana , which enables us to load a image from remote database so easily.

Reference: *AvFoundation.framework* *AVMetadataMachineReadableCodeObject*

Session 6: remote database

We use mysql1.cs.clemson.edu as the remote server. The server is installed with LAMP. The remote server is used to store the QR code information and the building information. And administrator will manage the information to help the user get the right QR code information from the server side, including images, titles and detailed descriptions. The building's information will all be uploaded by the user group. User can customize their places and they can also send it to the server side.

The schema of the table shows below:

```
create table if not exists QRcode
(
   QRId varchar(100) not null primary key,
   title varchar(50) not null,
   picPath varchar(200) not null,
   detail varchar(500) not null,
   weblink varchar(200),
   gps varchar(500)
)ENGINE=InnoDB;
```

```
create table if not exists building
(
  buildingId int(10) not null auto_increment primary key ,
  name varchar(100) not null,
  info varchar(500) not null,
  address varchar(500),
  tel varchar(10),
  picPath varchar(200),
  longtitude decimal(15,10) not null,
  latitude decimal(15,10) not null
)ENGINE=InnoDB;

create table if not exists nearby
(
  nearbyID int(10) not null auto_increment primary key,
  buildingId int(10) not null references building(buildingId),
  QRId varchar(100) not null references QRcode(QRId),
  distance varchar(5) not null,
  foreign key(buildingId) references building(buildingId),
  foreign key(QRId) references QRcode(QRId)
)ENGINE=InnoDB;
```

QRId is a primary key to identify the QR code. The picPath is an url of the picture(on the remote server). The detail depicts the detail information of the QR code. Also the buildingId is the primary key for the table building.

Reference: *AFNETWORKING*

Session 7: local database

The local database is a sqlite3 database. The schema is just the same as the remote one. As it is used as a cache. Each time user finds some QR code information on the remote server, the information is recoded in the local server. As we believe user will probably search for the same things in a short time. So each time the user tries to search something. The app will search the local database first and it is not there ,then the app will search the remote database. In this way, the cost is much smaller.

And the user can also synchronize from the remote database to the local database to get more information.

The schema shows below:

```
CREATE TABLE QRcode(QRId TEXT PRIMARY KEY NOT
NULL, title TEXT not null,picPath TEXT not
null,detail TEXT not null,gps TEXT)
```

```
CREATE TABLE building(buildingId INTEGER
PRIMARY KEY AUTOINCREMENT,name TEXT NOT
NULL,info TEXT NOT NULL,address TEXT,tel
TEXT,picPath TEXT,longtitude DECIMAL(15,10) NOT
NULL,latitude DECIMAL(15,10) NOT NULL)
```

```
CREATE TABLE nearby(nearbyID INTEGER primary
key AUTOINCREMENT,buildingId INTEGER not null
references building(buildingId),QRId INTEGER
not null references QRcode(QRId),distance TEXT
not null,foreign key(buildingId) references
building(buildingId),foreign key(QRId)
references QRcode(QRId))
```

Reference: *sqlite*