我的冰川

(一)　项目概述

我们的 APP 名称是“我的冰川”，每一个人一开始都有一块小小的“冰川”，通过步数统计、任务设置、游戏问答，使得界面发生变化，呼吁用户了解“冰川融化”的现状，掌握低碳绿色的生活，从而达到减缓南极冰川融化的目的。这是一款面向热爱环保的用户的环保教育类的 APP。

（左为设计 LOGO）

(二) 项目摘要

i.　灵感来源

温室效应不断积累，温度上升，全球气候变暖。全球变暖不仅危害自然生态系统的平衡，还威胁人类的生存。减少热量排放应放在每个人的日常生活习惯上。因此我们设计了以低碳日常习惯打卡“拯救冰川”的形式来呼吁用户拥有低碳绿色的生活方式。

ii.　市场调研

目前市场上的环保类 APP:

1.游戏教育类——通过游戏学习环保知识　（多为垃圾分类）

2.监测工具类——数据监测和分析

3.生活社交类——环保知识交流、环保材料购物等实际生活应用。

我们的 APP 属于生活类，但我们的主题是“冰川融化”。

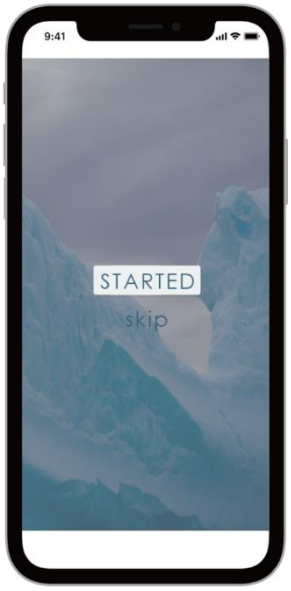
iii.　项目内容

提供日常打卡，游戏问答，步数记录使虚拟的冰川场景发生可视化变化，用户获得相应的虚拟奖励。

iv.　实现方法（简要设计思想及部分代码）

1.欢迎界面

第一次启动 APP 时出现，视图包含两个按钮，点击 STARTED 启动 APP，SKIP 则退出返回手机主界面。

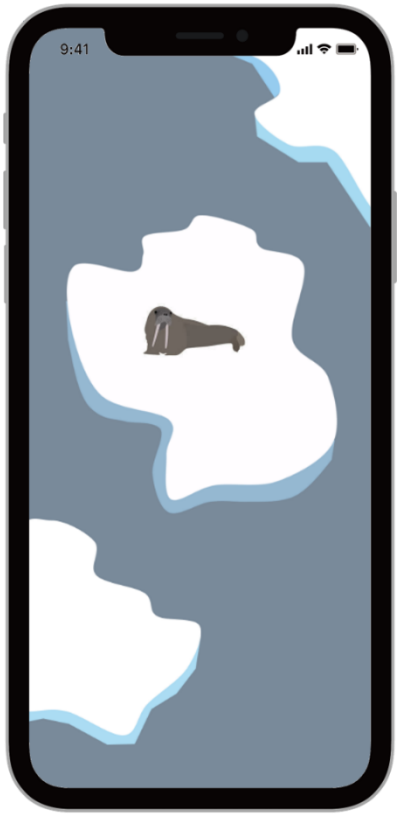



```
class HelloViewController: UIViewController {
@IBOutlet func StartbuttonTap(sender : UIButton){
    performSegueWithIdentifier("InitSegue",sender:nil)
}//当Started按钮被按下，跳转至初始化界面
@IBOutlet func SkipButtonTap(sender : UIButton){
    exit(0)
}//当Skip按钮被按下，退出程序
overrride func prepareForSegue(segue:UIStoryBoardSegue,
                               sender: AnyObject?){
    if segue.identifier == "InitSegue"{
        var destination: InitViewController =
        segue.destinationViewController as InitViewController
    }
}
```

2.初始化冰川

- 选择 STARTED 之后需要输入冰川名称(文本输入控件 UITextField，用户的输入内容（字符串），作为冰川名称)。
- 点击确认按钮进入到主界面。


```
@IBOutlet weak var NameField: UITextField!
NameField.placeholder = @"请输入冰川名称"
NameField.adjustsFontSizeToFitWidth=YES//设置调整文字大小以适配宽度
@IBAction func saveName(sender: UIButton){
    if(NameTextField.text != nil){
        Iceberg = IceBerg(Name:NameTextField.text)
        performSegueWithIdentifier("MainSegue",sender:nil)
    }//用户点击“保存”按钮后初始化Iceberg对象，跳转至主界面
}
override func prepareForSegue(segue: UIStoryBoardSegue,
                              sender: AnyObject?){
    if(segue.identifier == "MainSegue"){
        var destination: MainViewController =
            segue.destinationViewController as MainViewController
}
}
```



3.主界面

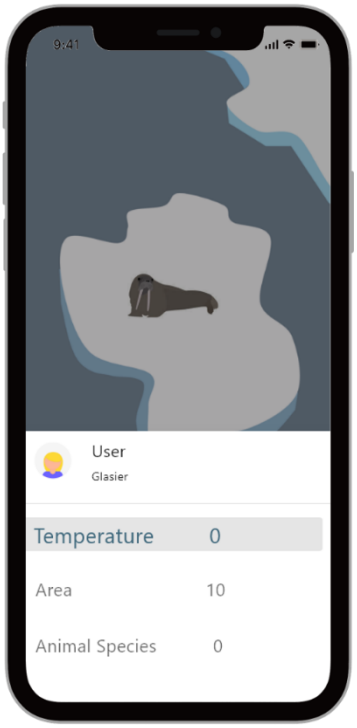
- 背景：View 上再添加一个 UIImageView 显示图片作为 UIView 的背景图片
- 冰川会根据用户数据的变化（成就点，为全局变量）进行可视化的改变，实现方法：设置梯度，到达某一梯度就改变冰川图片，图片资源拖放到 Images.xcassets 中存储，根据数据的变化响应放入视图。
- 动物图片资源与冰川图片资源存储和加载方式类似，UIImageView 控件用于显示图片，在冰川的显示范围内改变控件坐标以实现随机移动，图片视图中加入用户触碰交互，点击动物图片动物会自动移动。
- 使用 AVAudioPlayer 类添加背景音乐，导入 AVFoundation 框架，导入头文件#import <AVFoundation/AVFoundation.h>，导入背景音乐，代码中添加路径，创建播放器，设置播放

```
var IndexPic: Int = 1
if Score>=0 && Score < 50
   IndexPic = 1
else if Score >= 50 && Score < 100
   IndexPic = 2
else if Score >= 100 && Score < 200
   IndexPic = 3
//根据成就点改变图片序号
switch IndexPic {
   case 1  :
     iceImageView.image = UIImage(named:"Ice1")
   case 2  :
      iceImageView.image = UIImage(named:"Ice2")
   case 3  :
      iceImageView.image = UIImage(named:"Ice3")
}//根据序号显示不同的冰川图片
```

```
class Iceberg{
    var Name: String = ""//冰川名称
    var Temperature: Int = 0//冰川温度
    var Area: Int = 0//冰川面积
    var AnimalSpecies: Int = 0//动物种数

    //指定构造器
    init(Name: String,temperature: Int,Area: Int,AnimalSpecies: Int){
        self.Name = Name
        self.Temperature = Temperature
        self.Area = Area
        self.AnimalSpecies = AnimalSpecies
    }

    //便利构造器
    convenience init(Name: String){
        self.Init(Name: Name,Temperature: 0,Area: 10,AnimalSpecies: 0)
    }

    //描述方法，调试时查看对象信息
    func description() ->String{
        return "Name: \(Name) Temperature :\(Temperature)
        Area :\(Area) AnimalSpecies :\(AnimalSpecies)"
    }
```

4.参数界面

- **在主界面上滑操作时可查看冰川的各项以及用户名和冰川名称**
- **滑动实现：滚动视图(ScrollView)作为底层视图，响应滑动操作，加载的标签以及自定义的视图都贴到这个滚动视图上**



（MainViewController 中）

```
  let swipeUp = UISwipeGestureRecognizer(target:self,
                                         action:@selector(swipeGesture(_:)))
//创建上滑手势识别器
  swipeUp.direction = UISwipeGestureRecognizerDirection.Up//上滑
  self.view.addGestureRecognizer(swipeUp)

func swipeGesture(swip:UISwipeGestureRecognizer) {
    if swip.direction == UISwipeGestureRecognizerDirection.Up {
        performSegueWithIdentifier("DataSegue",sender:nil)
    }
}//上滑手势识别后,出现数据查看界面
@override func prepareForSegue(segue: UIStoryBoardSegue,sender :AnyObject?){
    if(segue.identifier == "DataSegue"){
        var destination: DataViewController =
                         segue.destinationViewController as SettingViewController
    }
}
```

（DataViewController 中）

```
  let swipeDown = UISwipeGestureRecognizer(target:self,
                                           action:@selector(swipeGesture(_:)))
//创建下滑手势识别器
  swipeDown.direction = UISwipeGestureRecognizerDirection.Down//下滑
  self.view.addGestureRecognizer(swipeDown)

func swipeGesture(swip:UISwipeGestureRecognizer) {
    if swip.direction == UISwipeGestureRecognizerDirection.Down {
        dismissViewControllerAnimated:YES completion:nil
    }
}//下滑手势识别后，销毁当前环境，回到主界面
```

5.功能选择

- 主界面左滑出现功能选择界面
- 五个选项：表格视图（TabelView）
- 点击头像(图片控件)，给图片控件添加点击动作，响应点击事件，进入用户信息设置：
- 头像设置从相册加载图片文件：

首先是获取到相册，在选择图片加载，相册资源访问通过 UIImagePickerController 类来读取，该类继承自 UINavigationController，是个地理的导航控制器，一般使用模态窗口的方式弹出。

- 修改冰川名称和用户名采用文本输入控件 UITextField。

```
@IBOutlet weak var Choosetableview: UITableView!
    var ChoiceSourse = ["任务打卡","运动计步","个人徽章","低碳日志","设置"]
    func ChooseTableView(tableView: UITableView, cellForRowAtIndexPath IndexPath:
                         NSIndexPath) -> UITableViewCell{
        var cell = UITableViewCell(style: UITableViewCellStyle.Default,
                                   reuseIdentifier: "Cell")
        if indexPath.row < ChoiceSourse.count {
            cell.setCellData(string: ChoiceSourse[indexPath.row] as! String)
        }
        return cell
    }//初始化单元格对象并返回所创建单元格对象
```

```
@IBOutlet var Headimg: UIImageView!
    override func viewDidLoad() {
        super.viewDidLoad()
        let imgClick = UITapGestureRecognizer(target: self, action:
                                              #selector(tapAction))
        Headimg.addGestureRecognizer(imgClick)
        //开启 isUserInteractionEnabled 手势否则点击事件会没有反应
        Headimg.isUserInteractionEnabled = true
    }

 //点击头像图片控件后跳转至设置界面
 @objc func tapAction() -> Void {
        performSegueWithIdentifier("SettingSegue",sender:nil)
 }
override func prepareForSegue(segue: UIStoryBoardSegue,sender: AnyObject?){
    if(segue.identifier == "SettingSegue"){
        var destination: SettingViewController =
            segue.destinationViewController as SettingViewController
        }
    }
}
```

```
UIImagePickerControllerDelegate,UINavigationControllerDelegate
let ActionSheet = UIAlertController.init(title: nil, message: nil,
                                                    preferredStyle: .actionSheet)
    weak var weakSelf = self
    let NanAction = UIAlertAction(title: "从相册中选择",  style: UIAlertActionStyle
        (action:UIAlertAction)in
              weakSelf?.initPhotoPicker()
         //填写需要的响应方法

        }

        let NvAction = UIAlertAction(title: "拍照", style: UIAlertActionStyle.defau
                     (action:UIAlertAction)in
         weakSelf?.initCameraPicker()
            //填写需要的响应方法

        }
        ActionSheet.addAction(NanAction)
        ActionSheet.addAction(NvAction)
        self.present(ActionSheet, animated: true, completion: nil)

    //从相册中选择
    func initPhotoPicker(){
        let photoPicker =  UIImagePickerController()
        photoPicker.delegate = self
        photoPicker.allowsEditing = true
        photoPicker.sourceType = .photoLibrary
        //在需要的地方present出来
        self.present(photoPicker, animated: true, completion: nil)
    }

    //拍照
    func initCameraPicker(){
        if UIImagePickerController.isSourceTypeAvailable(.camera){
            let  cameraPicker = UIImagePickerController()
            cameraPicker.delegate = self
            photoPicker.allowsEditing = true
            photoPicker.sourceType = .photoLibrary
            //在需要的地方present出来
            self.present(photoPicker, animated: true, completion: nil)
        }

        //拍照
        func initCameraPicker(){
            if UIImagePickerController.isSourceTypeAvailable(.camera){
                let  cameraPicker = UIImagePickerController()
                cameraPicker.delegate = self
                cameraPicker.allowsEditing = true
                cameraPicker.sourceType = .camera
                //在需要的地方present出来
                self.present(cameraPicker, animated: true, completion: nil)
            } else {
               print("不支持拍照")
            }
       func imagePickerController(picker: UIImagePickerController,
                                  didFinishPickingMediaWithInfo info: [String : Any]){
          //获得照片
          let image:UIImage = info[UIImagePickerControllerEditedImage] as! UIImage

          // 拍照
          if picker.sourceType == .camera {
           //保存相册
          UIImageWriteToSavedPhotosAlbum(image, self,
                 @selector(image(image:didFinishSavingWithError:contextInfo:)), nil)
          }
          headImage.image = image
          self.dismiss(animated: true, completion: nil)
       }
```

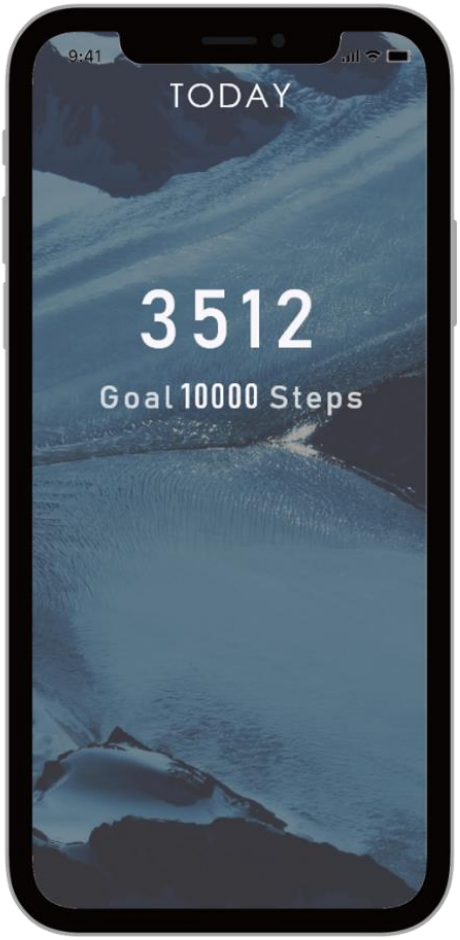
6.任务打卡

- 设置今日任务：点击加号随机生成任务和用户自定义
  -创建任务：TableView，显示图片和文字。
  -随机：默认任务的字符串数组，利用随机函数生成随机数作为序号抽取字符串，图片地址字符串数组与文字字符串数组对应，按照抽取的序号对任务标签进行设置。
  -用户自定义：文本框输入，添加图片需获取读取相册资源以及相机拍摄的权限。
- 任务打卡：点击完成图标，完成打卡
- 获取当前日期转换为星期几（NSDate），并且获取整个星期的日期，星期一重新获取


```
var missionArr  = String[]()
var missionSourse = ["垃圾分类","乘坐公交","使用环保材料","购买绿色产品","自行车出行"]

func randomMission() ->Void{
    let NwIntNumber = Int.randomIntNumber(lower: 0, upper: missSourse.count)
    for i in 0..<missionArr.count {
        if(missionArr[i] != missionSourse[NwIntNumber])//检验是否重复
            missionArr.append(missionSourse[NwIntNumber])
    }
}
//产生任务资源数组长度范围之内的随机数,将对应字符串放入任务数组
func dayOfWeek() -> Int {
        var interval = self.timeIntervalSince1970;
        var days = Int(interval / 86400);
        return (days - 3) % 7;
 }//获取当前日期并判断为星期几
```

7.个人徽章

按照垃圾分类答题，每日计步，任务打卡的数据设置响应的成就，可左右滑动（主视图里面添加几个成就视图，添加 swipe 手势，左右滑动时，响应事件处理）浏览，获得新成就时在主界面会出现消息提醒（UIAlertView 消息提示框）。



8.运动计步

- 获取每日步数：在 info.plist 添加权限，以便访问 HealthKit，开启健康数据中心，创建 healthStore 实例对象，从健康应用中获取读取步数的权限，查询步数的数据，并且以字符串形式显示在界面上，将步数的数据按照给定的能量点换算的算法换算成能量点，能量点会影响主界面的冰川可视化形态和冰川的参数。
- 设置目标步数（达到目标可以获得额外成就点）：用户输入目标步数（文本框输入）以字符串形式显示在界面上，验证到达目标时与实际步数进行比对，如果达到目标则增加成就点。

9.低碳日志

- 用户点击空白框进行输入并显示：

使用 UITextView，可显示多行文本视图，接受编辑输入，类似于文本编辑器，获取焦点的时候会从底部弹出软键盘输入。

- 显示日期：NSDate 获取当前日期，显示当且月份并且将当前日期换算当前月份占比百分制显示。
- 随机出现主题：今日****，文本框默认显示字符，用户点击即可编辑新内容
- 打卡内容中的任务字符串自动导入日志中并显示

10.垃圾分类

- 创建垃圾类，封装垃圾类别和名称以及其他信息，创建垃圾对象数组，问题中随机出现。
- 四个种类为按钮，出现问题对话框时动态产生，用户点击时响应事件内读取按钮上的字符串与问题中的垃圾对象的种类字符串进行比对，如果一致则正确，增加成就点；错误则不作操作。
- 用户点击确定按钮时出现结果，退出则返回主界面。

```
class Rubbish{
    var Name: String =""
    var Type: String =""

    init(Name: String,Type: Type){
        self.Name = Name
        self.Type = Type
    }
}//Rubbish类
```

```
var AnswerArr = ["可回收垃圾","其他垃圾","厨余垃圾","有害垃圾"]
// button数组
    var buttonArr:NSMutableArray = []
    // 创建view方法
    func creatLineButton(dataArr:NSMutableArray,buttonSize:CGSize) ->UIView {
        for index in 0..<AnswerArr.count {
            // 定义button
            let button = UIButton(type: UIButtonType.System)
            button.setTitle(AnswerArr[index] as? String)
            button.tintColor = UIColor.whiteColor()
            button.backgroundColor = UIColor.lightGrayColor()
            button.titleLabel?.font = UIFont.systemFontOfSize(20)
            // 点击方法
            button.addTarget(self, action: Selector("buttonClick:"),
                             forControlEvents: UIControlEvents.TouchUpInside)
            // 添加到view上
            QustionView.addSubview(button)
            // 加入button数组
            buttonArr.addObject(button)
        }
```

```
func buttonClick(button:UIButton){
    for b in buttonArr{
        if (b as! UIButton) == button{
            (b as! UIButton).backgroundColor = UIColor.blueColor()
            if((b as! UIButton).Title == Type){
                Score++;
                let alert = UIAlertController(title: "提示",
                    message: "恭喜你答对了",
                    preferredStyle: UIAlertControllerStyle.Alert)
                let acSure = UIAlertAction(title: "确定",
                    style: UIAlertActionStyle.Destructive) {
                    (UIAlertAction) -> Void in    print("click Sure")
            }
                alertVC.addAction(acSure)
                self.presentViewController(alertVC, animated: true,
                                           completion: nil)
            }else{
                let alert = UIAlertController(title: "提示",
                    message: "抱歉回答错误",
                    preferredStyle: UIAlertControllerStyle.Alert)

              let acSure = UIAlertAction(title: "确定",
                 style: UIAlertActionStyle.Destructive) {
                  (UIAlertAction) -> Void in print("click Sure")
        }
                alertVC.addAction(acSure)
                self.presentViewController(alertVC, animated: true, completion: nil)
            }
    }
}
```

v. 目标效果

使用户可以意识到“冰川融化”背后隐藏的温室效应的危害性，从而呼吁用户拥有环保意识，并且拥有健康低碳的生活方式。

(三) 项目亮点

i. **用户自定义的虚拟场景——我的冰川，通过可视化的方式让用户体验到冰川融化的过程。**

用户拥有自己的一片冰川地带，用户可以自定义冰川的名字。显示冰川的虚拟场景（初始冰川面积最小，温度最高，融化速度最快，无极地动物出没）

ii. **用户通过随机生成或者自定义的任务进行每日的生活打卡，打卡效果反馈到虚拟场景中。**

系统（可以随机得到低碳任务）给出低碳出行的任务清单，完成即可打卡，打卡后获得相应的成就点。

- 成就点增加可以使冰川面积扩大，温度下降，融化速度降低（可视化）。
- 成就点积累到某一数量即可解锁新的动物种类，解锁的动物会出现在我的冰川的界面上，并获得成就徽章。
- 计步器记录每天步行步数，步数达到所设置标准值，相应成就点增加
- 用户可以自定义打卡内容，规定打卡的目标天数

- 如果没有及时打卡或者步数未达到标准冰川面积缩小，温度上升，融化速度加快

iii. 用户的低碳行动记录自动生成“低碳日历”。（用户也可自定义编辑）
iv. 随机出现海底垃圾用户通过问答方式选择种类。（融合垃圾分类知识）

(四) 前景和价值

互联网 + 环保

“互联网+”浪潮来袭，公开、共享成为大势所趋，“互联网+”意味着环保将更方便、更时尚，环保将成为更多人的生活态度，而环保类 APP 正是这种互联网和环保事业结合的最好体现。

环保知识普及

现今很多人并不了解冰川融化和冰川污染的严重性和危害性，气候变化的影响往往是不可逆的，而我们的 APP 可以引起人们对于冰川融化问题的关注以及提高对环保意识的普及。

全民低碳生活

通过本 APP 增强人们的低碳环保意识，垃圾分类的意识，使得全球的环境问题能有所缓解。