Logiciels embarqués ambiants/iOS

Chapitre 4 : User Inputs

Dr. Abdelkader Gouaïch[1]

[1] Department of Computer Science
Université de Montpellier

2012

Outline

Outline

Objetifs de ce chapitre

- Nous allons voir aujourd'hui la gestion des entrées d'utilisateur (User Input)
- Ecran tactile
- Accéléromètre

Résumé de la séance précédente

- La semaine dernière nous avons présenté la gestion du rendu graphique avec OpenGL ES.
- Nous reviendrons plus tard sur l'aspect graphique pour présenter plus en détails (selon le temps et la motivation) les points suivants :
  - SpriteSheet (Packed) : comment utiliser une seule feuille (texture) pour regrouper plusieurs sprites
  - TileMap : comment construire de grandes cartes simplement en composant des briques de base : tiles
  - Font : comment fabriquer de nouvelles fonts pour écrire du texte
  - Animation : comment faire une animation 2D avec une SpriteSheet

- Cependant, nous sommes capables aujourd'hui de comprendre le fonctionnement de toutes ces classes à partir du chapitre consacré aux textures et aux images.
- En effet, ces fonctionnalités ne sont que l'utilisation directe de ce que nous avons présenté.

SpriteSheet

- SpriteSheet est simplement une texture composée de plusieurs sous-images.
- Packed SpriteSheet est aussi une texture composée de plusieurs sous-images avec des dimensions différentes donc un autre fichier de description est nécessaire pour retrouver les sous-images.

TileMap

- TileMap : Nous allons simplement charger les tiles comme texture et lire la description de la tileMap afin de bien générer les bonnes images

Font

- Font : Nous allons produire une texture qui contient tous les caractères de la Font et un fichier de description pour faire le lien entre les caractères et les sous-images. Pour écrire un texte, nous allons dessiner les caractères qui le composent.

Animation

- Animation : Texture + gestion du temps. Nous allons introduire une gestion du temps pour savoir quel sous-images et pendant combien de temps.

TileMap

- Ces fonctionnalités vont être disponibles à partir d'aujourd'hui pour vous dans le moteur pour votre projet de jeu.

iOS et user input

- Nous allons présenter aujourd'hui ces deux mécanismes pour recueillir les inputs de l'utilisateur (user input)
- IPhone a introduit deux mécanismes d'interaction avec l'utilisateur assez inédits : l'accéléromètre et le multi-touch (sans stylet)
- A nous d'imaginer de nouvelles formes d'interaction avec ces fonctionnalités car leur utilisation est vraiment très simple !

Multi touch

- Quand l'utilisateur va toucher l'écran iOS va produire des événements qui vont nous permettre de savoir :
- Type d'événement (touché, télécommande)
- la localisation du touch
- le mouvement/geste

Accéléromètre

- L'accéléromètre permet de mesurer l'accélération subit par l'appareil (x,y,z)

- Deux types d'événements sont proposés :
  - Touch event
  - Motion event
- Les deux événements héritent de la classe UIEvent
- iOS va continuellement créer dans sa boucle applicative les évenements et les soumettre aux listeners enregistrés.

- Pour le traitement des événements voici les méthodes :
- touchesBegan :withEvent :
- touchesMoved :withEvent :
- touchesEnded :withEvent :
- touchesCancelled :withEvent :
- Pour pouvoir observer et répondre aux événements de touch/motion il nous faut faire partie de la chaine de traitement (UIResponder) et realiser au moins une de ces méthodes.

- Pour le traitement des événements voici les méthodes :
- touchesBegan :withEvent :
- touchesMoved :withEvent :
- touchesEnded :withEvent :
- touchesCancelled :withEvent :
- Pour pouvoir observer et répondre aux événements de touch/motion il nous faut faire partie de la chaine de traitement (UIResponder) et realiser au moins une de ces méthodes.

- Les méthodes présentées sont appelées durant les phases suivantes (touch) :
  1. Je touche l'écran
  2. Je bouge mon doigt
  3. Je lève mon doigt et je ne touche plus l'écran

- Les méthodes présentées sont appelées durant les phases suivantes (touch) :
  1. Je touche l'écran
  2. Je bouge mon doigt
  3. Je lève mon doigt et je ne touche plus l'écran

- Les méthodes présentées sont appelées durant les phases suivantes (touch) :
  1. Je touche l'écran
  2. Je bouge mon doigt
  3. Je lève mon doigt et je ne touche plus l'écran

activation du multitouch

- Dans le cas où le multitouch est activé alors ces phases sont propres à chaque doigt (ils peuvent être parallèle)
- Le premier paramètre est NSSet qui contient des UITouch et UIEvent type associé (touchn, motion, télécommande)

```
-(void)touchesBegan:(NSSet*)touches withEvent:(
        UIEvent*)event}
```

touchesMoved/touchesEnded/touchesCancelled

- Si le touch a été identifié et qu'il bouge alors la méthode sera (touchesMoved) appelée.
- Quand le touch se termine et bien la méthode touchesEnded est appelée.
- touchesCancelled est appelée en cas d'annulation de l'action : réception d'un coup de téléphone, sms etc.

Outline

- La classe qui fait partie de la chaine de récéption des événements est la classe EAGLView
- Nous allons utiliser la délégation pour transmettre les événements à un objet délégué : par exemple la scène courante

```
- (void)touchesBegan:(NSSet*)touches withEvent:(UIEvent*)event {
[[sharedGameController currentScene] touchesBegan:touches withEvent:event
view:self];
}
- (void)touchesMoved:(NSSet*)touches withEvent:(UIEvent*)event {
[[sharedGameController currentScene] touchesMoved:touches withEvent:event
view:self];
}
- (void)touchesEnded:(NSSet*)touches withEvent:(UIEvent*)event {
[[sharedGameController currentScene] touchesEnded:touches withEvent:event
view:self];
}
- (void)touchesCancelled:(NSSet*)touches withEvent:(UIEvent*)event {
[[sharedGameController currentScene] touchesCancelled:touches
withEvent:event view:self];
}
```

Outline

Traitement des événements

- Nous savons comment récupérer l'événement donné par iOS
- Nous voir maintenant comment l'exploiter/traiter
- Exemple d'illustration : nous allons créer un joystick virtuel

Récupérer la position du touch

- La classe UITouch possède la méthode locationInView : qui nous donne les coordonnées Il est important de donner une instance de UIView pour calculer les coordonnées par rapport à la vue actuelle
- Exemple : vue portrait/paysage

La phase touchesBegan

```
- (void)touchesBegan:(NSSet*)touches withEvent:(UIEvent*)event
view:(UIView*)aView {
for (UITouch *touch in touches) {
CGPoint originalTouchLocation = [touch locationInView:aView];
CGPoint touchLocation = [sharedGameController
adjustTouchOrientationForTouch:originalTouchLocation];
if (CGRectContainsPoint(joypadBounds, touchLocation) &&
!isJoypadTouchMoving) {
isJoypadTouchMoving = YES;
joypadTouchHash = [touch hash];
break;
}
}
}
```

Remarques/Commentaires

- Remarque : il nous faire une transformation des coordonnées de UIKit (iOS) et OpenGL
- OpenGL ES y commence en bas de l'écran 0 (botton) à 480 (top) – iphone
- UIView le y comment en haut de l'écran 0 (top) à 480 (bottom) – iphone
- Le role du hash est d'identifier de façon unique une instance de touch que nous allons suivre
- Il faut activer explicitement le multitouch dans UIView par la function

```
[ glView setMultipleTouchEnabled:YES ] ;
```

Remarques/Commentaires

- Remarque : il nous faire une transformation des coordonnées de UIKit (iOS) et OpenGL
- OpenGL ES y commence en bas de l'écran 0 (botton) à 480 (top) – iphone
- UIView le y comment en haut de l'écran 0 (top) à 480 (bottom) – iphone
- Le role du hash est d'identifier de façon unique une instance de touch que nous allons suivre
- Il faut activer explicitement le multitouch dans UIView par la function

```
[ glView setMultipleTouchEnabled:YES ] ;
```

La phase touchesMove

```
- (void)touchesMoved:(NSSet*)touches withEvent:(UIEvent*)event view:(UIView*)aView {
for (UITouch *touch in touches) {
if ([touch hash] == joypadTouchHash && isJoypadTouchMoving) {
CGPoint originalTouchLocation = [touch
locationInView:aView];
CGPoint touchLocation = [sharedGameController
adjustTouchOrientationForTouch:originalTouchLocation];
float dx = (float)joypadCenter.x - (float)touchLocation.x;
float dy = (float)joypadCenter.y - (float)touchLocation.y;
// Manhattan Distance
joypadDistance = abs(touchLocation.x - joypadCenter.x) +
abs(touchLocation.y - joypadCenter.y);
directionOfTravel = atan2(dy, dx);
}
}
}
```

Commentaires

- Les deux attributs directionOfTravel et joypadDistance sont utilisés dans updateSceneWithDelta : pour dessiner la position du personage

La phase touchesEnded

- (void)touchesEnded :(NSSet*)touches withEvent :(UIEvent*)event view :(UIView*)aView for (UITouch *touch in touches) if ([touch hash] == joypadTouchHash) isJoypadTouchMoving = NO ; joypadTouchHash = 0 ; directionOfTravel = 0 ; joypadDistance = 0 ; return ;

Outline

Compter le nomber de tapes

- L'attribut tapCount de UITouch donne le nombre de tapes

```
if (touch.tapCount == 2) {
//faire qq chose ici
}
```

Outline

Initialiser l'accéléro

- setUpdateInterval : fréquence d'appel du délégué (100 par secondes par exemple)
- setDelegate : permet de designer l'objet délégué

```
[[UIAccelerometer sharedAccelerometer]
        setUpdateInterval:1.0 / 100.0];
[[UIAccelerometer sharedAccelerometer] setDelegate:
        currentScene];
```

Protocole IAccelerometerDelegate

- Le délégué doit se conformer au protocole : IAccelerometerDelegate

```
- (void)accelerometer:(UIAccelerometer *)accelerometer didAccelerate:(UIAcceleration *)
            acceleration {
    // This is a very basic low-pass filter
accelerationValues[0] = acceleration.x * 0.1f + accelerationValues[0] * (1.0 - 0.1f);
accelerationValues[1] = acceleration.y * 0.1f + accelerationValues[1] * (1.0 - 0.1f);
accelerationValues[2] = acceleration.z * 0.1f + accelerationValues[2] * (1.0 - 0.1f);
}
```

updateScene

```
- (void)updateSceneWithDelta:(float)aDelta {
if (!sharedGameController.eaglView.uiSwitch.on) {
knightLocation.x -= (aDelta * (20 * joypadDistance)) *
cosf(directionOfTravel);
knightLocation.y -= (aDelta * (20 * joypadDistance)) *
sinf(directionOfTravel);
}
if (sharedGameController.eaglView.uiSwitch.on) {
knightLocation.x += aDelta * (accelerationValues[0] * 1000);
knightLocation.y += aDelta * (accelerationValues[1] * 1000);
}
if (knightLocation.x < 0)
knightLocation.x = 0;
if (knightLocation.x > 320)
knightLocation.x = 320;
if (knightLocation.y < 0)
knightLocation.y = 0;
if (knightLocation.y > 480)
knightLocation.y = 480;
}
```

Objectifs du TP

- Consulter les classes suivantes dans le code du projet CH12SQLTSOR :
  - GameScene
  - GameControler
  - EAGLView
  - AbstractScene
- Votre objectif est de reprendre le TP précédant avec l'image qui rebondit sur les bords de l'écran.
- Modifier votre code pour commencer l'animation de l'image et l'arrêter quand l'utilisateur tape dessus