Introduction Exercise

(Android game application)

**Introduction**

The goal of this introduction exercise is to get you familiarized with the Android.

(**Sheep** is an extended library based on Android platform, making android to be a “XNA-like” game studio, it provides reusable components that can be convenient to use in game development, short your game development time. It is an optional library for you to use. It could run on r04 and r08 version, but has not been strictly testing on the newest r08 SDK android platform.)

**Delivery**

You should **zip and deliver all the codes and files under your project folder together with a .apk file.** (If you think sheep lib is useful, you can choose to develop your game application with Sheep, it is not mandatory.)

**TASK 1 – Sprites**

For this task, you should draw graphics on the screen (You can use Sprite class from Sheep). You can use the helicopter texture or your own texture for the sprite.



Draw the helicopter on the screen and make it move around on its own. If the sprite is about to leave the screen, it should “bounce” off the edges and head in the opposite direction. Also, make sure the helicopter faces the direction it is traveling.

**TASK 2 – Input and text**

Reuse the helicopter sprite from task 1.

a) Draw the sprite on the screen, but instead of having it go around on its own, make it so that you can use touch function to control the movement. The sprite must not be allowed to exit the screen.

b) Print the position of the sprite (in screen coordinates). The text should be drawn in the upper-left corner of the screen.

**TASK 3 – Animation, timing, and collision detection**

In this task, use the helicopter sprite sheet. The sprite sheet contains the animation frames for the helicopter, simulating the spinning of the rotor blades. Each frame is 130 pixels wide and 52 pixels high.



a) Create the same scenario as in task 1, but apply animation to the sprite. Each frame in the animation should display for 100 ms before changing to the next frame. Also, the animation should loop, meaning that you start around again with the first frame after the last one finished.

b) Throw in a couple more sprites (you can use the same sprite, or create new ones yourself). Move the sprites around in random directions and with random speed. If the sprites collide, i.e. their bounding volumes intersect, the sprites should bounce off each other.

**TASK 4 – Create Pong**

Build on what you have learned in the previous tasks, and implement your own version of the classic game Pong. The game should follow the following basic rules, but feel free to apply your own twists to it:

- There are two paddles, one on each side of the screen. The paddle can be moved vertically and should be kept inside the screen. It is up to you whether both paddles should be controlled by humans (multiplayer), or one of the paddles should be controlled by the computer (single player).

- The ball should start off in the center of the screen and bounce off the paddles, and the upper and lower edges of the screen.

- If the ball slips past a paddle, the paddle on the opposite side is given one point, and the ball is reset to the middle of the screen. The score should be displayed somewhere on the screen. First one to reach 21 wins the game.

Hints and tips:

- Experiment with the speed of the ball and paddles to make the game challenging. It should not be too easy to reach the ball; neither should it be too hard.

- To make things even more challenging, try increasing the speed of the ball after a certain time if no one scores a point, or if the ball hits the edges of the paddles

