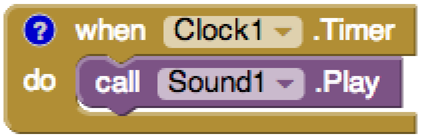
How do you code your app to do something repeatedly, pausing in between each iteration?

How do you program time? How do you program animation? With App Inventor, the Clock component can serve as an alarm clock, and its Timer event can trigger activity. Check out the following examples:

Example 1. How do you play a sound every second?

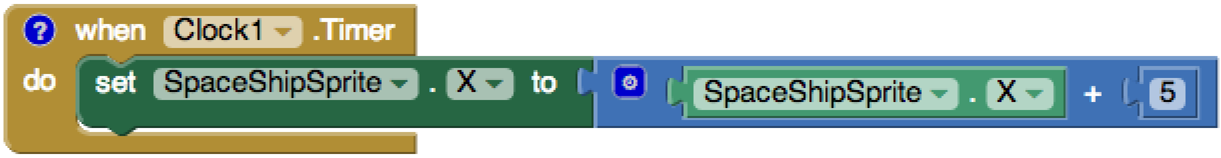


```
when Clock1 .Timer
do  call Sound1 .Play
```

The when Clock.Timer event is triggered repeatedly. Think of it like an alarm clock. The Clock.TimerInterval property, determines how often the sound will play. By default it is 1000 milliseconds (1 second), so if you don't change it the sound above will play every second. You can set this property in the designer or with blocks.

Question: How can you play a sound every five seconds instead of every second? answer

Example 2. How do you move a spaceship smoothly across the screen, i.e., how does animation work?



```
when Clock1 .Timer
do  set SpaceShipSprite . X to   SpaceShipSprite . X + 5
```

The spaceship's X coordinate is its horizontal location. On each timer event, the X coordinate is increased and the spaceship appears further to the right.

If the Clock.TimerInterval is set to the default of 1000 ms, then the ship will move every one second, a very choppy motion. Typically, for this type of animation, you'll set the timer interval to something like 40 ms.

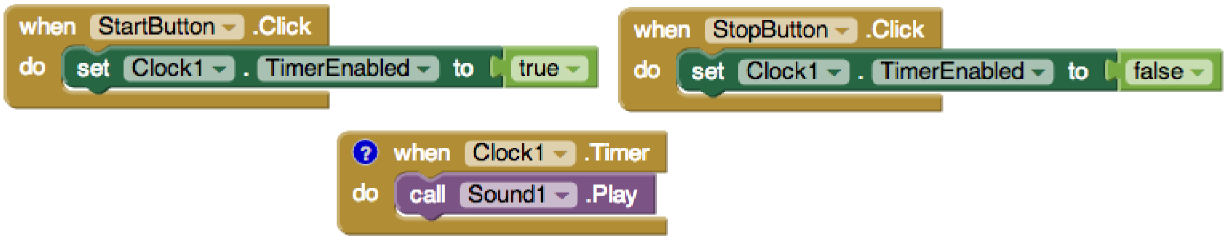
Question: Moving pictures-- animation-- is really displaying a sequence of pictures very fast. Moving pictures have a frame rate--how many pictures (frames) are shown every second. If you set the timer interval for the spaceship to 40 ms, what will be the frame rate for your "movie"? What is the typical frame rate for a movie you see at the theater? answer

Note: for sprite movement, you can use the sprite's internal clock instead of a separate clock component.

How do you start and stop an animation?

Just like you can turn an alarm clock on and off, you can turn the Clock component's timer on and off. Check out the following examples:

Example 1. When the user clicks a start button, start playing a sound every second. When they click a stop button, stop playing the sound.

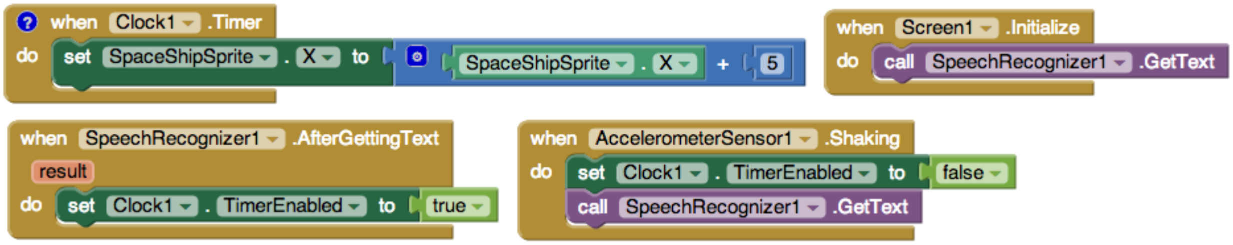


```
when StartButton .Click
do  set Clock1 . TimerEnabled to true

when StopButton .Click
do  set Clock1 . TimerEnabled to false

? when Clock1 .Timer
do  call Sound1 .Play
```

The clock has a Clock.TimerEnabled property. When it is true, the alarm clock is on so the timer will trigger. When Clock.TimerEnabled is set to false the timer is dormant.

When the user clicks the start button, TimerEnabled is set to true to get things moving. A second later, and each second thereafter, the when Clock.Timer event will trigger and the sound will play. When the user clicks the stop button, the timer is disabled and the sound stops playing.

Example 2. When the user speaks, start the spaceship moving. When the user shakes the phone, stop the spaceship from moving.



```
? when Clock1 .Timer
do  set SpaceShipSprite . X to ( SpaceShipSprite . X + 5 )

when Screen1 .Initialize
do  call SpeechRecognizer1 .GetText

when SpeechRecognizer1 .AfterGettingText
  result
do  set Clock1 . TimerEnabled to true

when AccelerometerSensor1 .Shaking
do  set Clock1 . TimerEnabled to false
    call SpeechRecognizer1 .GetText
```

You need two additional components, a SpeechRecognizer that can recognize when the user is speaking, and an AccelerometerSensor that senses when the device is being shook.

The Screen.Initialize event handler is triggered when the app begins. The SpeechRecognizer.GetText block opens the speech recognizer so that it is waiting for the user to say something. When the user says something, the SpeechRecognizer.AfterGettingText event handler is triggered. Here, you set Clock.TimerEnabled to true so the clock timer will start triggering and the sprite will begin moving. When the user shakes the phone, AccelerometerSensor.Shaking is triggered, the timer is disabled so the sprite stops, and the SpeechRecognizer.GetText is called to await the next time the user speaks.