

Moving around



You can use the arrow keys – or other keys like a, d, w, s – to move around an asset.

In this example we are focusing on the movements and we use an asterisk instead of an asset.

Create a `main.lua` file that:

- Create a star table (`star = { ... }`) containing the x and y coordinates, the movement speed and the character to be shown (the * asterisk).
- Create an `update(dt)` function, that changes the x and y coordinates depending on the key being pressed.
- In `draw()`, draw the star character at the x and y coordinates.

speed is the number of pixel to move per second. The `dt` argument in `update(dt)`, allows us to know how much of this distance should be moved on each call of `update()`.

Moving around



You can use the arrow keys – or other keys like a, d, w, s – to move around an asset.

In this example we are focusing on the movements and we use an asterisk instead of an asset.

Create a `main.lua` file that:

- Create a star table (`star = { ... }`) containing the x and y coordinates, the movement speed and the character to be shown (the * asterisk).
- Create an `update(dt)` function, that changes the x and y coordinates depending on the key being pressed.
- In `draw()`, draw the star character at the x and y coordinates.

speed is the number of pixel to move per second. The `dt` argument in `update(dt)`, allows us to know how much of this distance should be moved on each call of `update()`.

Moving around



You can use the arrow keys – or other keys like a, d, w, s – to move around an asset.

In this example we are focusing on the movements and we use an asterisk instead of an asset.

Create a `main.lua` file that:

- Create a star table (`star = { ... }`) containing the x and y coordinates, the movement speed and the character to be shown (the * asterisk).
- Create an `update(dt)` function, that changes the x and y coordinates depending on the key being pressed.
- In `draw()`, draw the star character at the x and y coordinates.

speed is the number of pixel to move per second. The `dt` argument in `update(dt)`, allows us to know how much of this distance should be moved on each call of `update()`.

Moving around



You can use the arrow keys – or other keys like a, d, w, s – to move around an asset.

In this example we are focusing on the movements and we use an asterisk instead of an asset.

Create a `main.lua` file that:

- Create a star table (`star = { ... }`) containing the x and y coordinates, the movement speed and the character to be shown (the * asterisk).
- Create an `update(dt)` function, that changes the x and y coordinates depending on the key being pressed.
- In `draw()`, draw the star character at the x and y coordinates.

speed is the number of pixel to move per second. The `dt` argument in `update(dt)`, allows us to know how much of this distance should be moved on each call of `update()`.

Since `update()` is being called many times per second the `dt` parameter is the fraction of second elapsed since the last call of `update()`.

```
star = {x = 175, y = 200, speed = 150, char = '*'}
```

```
function love.load(arg)
    love.graphics.setFont(love.graphics.newFont(36))
end
```

```
function love.update(dt)
    if love.keyboard.isDown('left', 'a') then
        star.x = star.x - (star.speed * dt)
    end
    if love.keyboard.isDown('right', 'd') then
        star.x = star.x + (star.speed * dt)
    end
    if love.keyboard.isDown('up', 'w') then
        star.y = star.y - (star.speed * dt)
    end
    if love.keyboard.isDown('down', 's') then
        star.y = star.y + (star.speed * dt)
    end
end
```

```
function love.draw()
    love.graphics.print(star.char, star.x, star.y)
end
```

Since `update()` is being called many times per second the `dt` parameter is the fraction of second elapsed since the last call of `update()`.

```
star = {x = 175, y = 200, speed = 150, char = '*'}
```

```
function love.load(arg)
    love.graphics.setFont(love.graphics.newFont(36))
end
```

```
function love.update(dt)
    if love.keyboard.isDown('left', 'a') then
        star.x = star.x - (star.speed * dt)
    end
    if love.keyboard.isDown('right', 'd') then
        star.x = star.x + (star.speed * dt)
    end
    if love.keyboard.isDown('up', 'w') then
        star.y = star.y - (star.speed * dt)
    end
    if love.keyboard.isDown('down', 's') then
        star.y = star.y + (star.speed * dt)
    end
end
```

```
function love.draw()
    love.graphics.print(star.char, star.x, star.y)
end
```

Since `update()` is being called many times per second the `dt` parameter is the fraction of second elapsed since the last call of `update()`.

```
star = {x = 175, y = 200, speed = 150, char = '*'}
```

```
function love.load(arg)
    love.graphics.setFont(love.graphics.newFont(36))
end
```

```
function love.update(dt)
    if love.keyboard.isDown('left', 'a') then
        star.x = star.x - (star.speed * dt)
    end
    if love.keyboard.isDown('right', 'd') then
        star.x = star.x + (star.speed * dt)
    end
    if love.keyboard.isDown('up', 'w') then
        star.y = star.y - (star.speed * dt)
    end
    if love.keyboard.isDown('down', 's') then
        star.y = star.y + (star.speed * dt)
    end
end
```

```
function love.draw()
    love.graphics.print(star.char, star.x, star.y)
end
```

Since `update()` is being called many times per second the `dt` parameter is the fraction of second elapsed since the last call of `update()`.

```
star = {x = 175, y = 200, speed = 150, char = '*'}
```

```
function love.load(arg)
    love.graphics.setFont(love.graphics.newFont(36))
end
```

```
function love.update(dt)
    if love.keyboard.isDown('left', 'a') then
        star.x = star.x - (star.speed * dt)
    end
    if love.keyboard.isDown('right', 'd') then
        star.x = star.x + (star.speed * dt)
    end
    if love.keyboard.isDown('up', 'w') then
        star.y = star.y - (star.speed * dt)
    end
    if love.keyboard.isDown('down', 's') then
        star.y = star.y + (star.speed * dt)
    end
end
```

```
function love.draw()
    love.graphics.print(star.char, star.x, star.y)
end
```