LAIG 20/21

TCCM

Class 5
(27/10/20)

PARSER

ANIMATION CLASS:

Animation with start and end state

constructor
- start time (seconds)
- end time (seconds)
- start transformations*
- end transformations*

KEYFRAME ANIMATION CLASS:

animation with ordered array of keyframes

constructor
- Array: keyframe [instant (seconds), transformations*]

The constructor functions must:
1. Save data passed as arguments
2. Initialize current state and other necessary variables
   - initial/last time
   - current state → matrix or vec3
   - ...

*Transformations correspond to a list of

Suggestion: vec3

Translation
Rotation X
Y
Z
Scale

UPDATE

In scene:
- setUpdatePeriod() when graph finished loading
- update(): function that triggers update of all animations

In Animation Class:

update(currentTime)

1. Check if animation is active → if not, return
2. Calculate animation's elapsed time → using deltaTime
3. Calculate current values for each transformation → if currentTime < endTime

$$X = X_i + X_{TOTAL} \times \frac{T_{ELAPSED}}{T_{TOTAL}}$$

($T_{ELAPSED}$ → time from start until now; T_{TOTAL} → total animation time)

- X: current value
- X_i: initial value
- X_{TOTAL}: value between initial/end time

Example: Updating Y value of translation at t = 8

startime = 4, endtime = 10, initial Y = 0, end Y = 6

$$Y = 0 + (6-0) \times \frac{(8-4)}{(10-4)} = 6 \times \frac{4}{6} = 4$$

($\frac{4}{6}$ → percentage of elapsed time)

4. Update current state and other variables ← matrix or vec3

In KeyframeAnimation Class:

Similar process to Animation.update() except for step 3

update(currentTime)

1. Check if animation is active → if not, return
2. Calculate animation's elapsed time → using deltaTime
3. Calculate current values for each transformation
   - 3.1. Get previous and next Keyframe
   - 3.2. For each value:

$$X = X_i + X_{TOTAL} \times \frac{T_{ELAPSED}}{T_{TOTAL}}$$

($T_{ELAPSED}$ → time from start until now; T_{TOTAL} → time of next keyframe)

- X: current value
- X_i: value in previous keyframe
- X_{TOTAL}: value between previous and next keyframe

Example: Updating Y value of translation at t = 8

KF0: instant = 0, Y = 0

KF1: instant = 4, Y = 2

KF2: instant = 12, Y = 6

previous = KF1; next = KF2

$$Y = 2 + (6-2) \times \frac{8-4}{12-4} = 2 + 4 \times \frac{4}{8} = 4$$

4. Update current state and other variables ← matrix or vec3

APPLY

During processNode() or display() of Node

1. Check if node has animation
2. Call animation.apply() after node matrix is applied
3. Continue processing/displaying descendants if animation is active

In Animation/KeyframeAnimation Class

apply()

Apply transformations to scene

If current state is vec3
- scene.translate
- scene.rotate x3
- scene.scale

Else current state is matrix
- scene.multMatrix

Note:

glMatrix provides functions for interpolation:

(static) lerp(out, a, b, t) → {vec3}

Performs a linear interpolation between two vec3's

Parameters:

Name	Type	Description
out	vec3	the receiving vector
a	ReadonlyVec3	the first operand
b	ReadonlyVec3	the second operand
t	Number	interpolation amount, in the range [0-1], between the two inputs