Basic concept of 2D game design and development

David Botzenhart, Kathrin Krisch

19th January 2007

Analysis

What makes a 2D Game?

Transformation of the Design Analysis into a Software Model.

Modeling

The Game class.

Animations and Sounds("The moving pictures")

Events and Collision Detection

Events

Collision Detection

Good Bye

The Idea

Beside the fact that a good "programming" is in not the only key to a successful game, we will focus on thinking of "how" to develop a game in a technical approach.
The most important thing ,of course is the Idea, closely followed by good design (audio and video) and a stable realization. Keep in mind that games are made for entertainment.
But also, games can be a source for powerful ideas...

Recognizable parts of a 2D game

What does appear in a game? Let us think about it...

- Screen.

Recognizable parts of a 2D game

What does appear in a game? Let us think about it...

- Screen.

- Sounds.

Recognizable parts of a 2D game

What does appear in a game? Let us think about it...

- Screen.
  - Frame or Status Bar.
- Sounds.

Recognizable parts of a 2D game

What does appear in a game? Let us think about it...

- Screen.
  - Frame or Status Bar.
  - Game Window.
- Sounds.

Recognizable parts of a 2D game

What does appear in a game? Let us think about it...

- Screen.
  - Frame or Status Bar.
    - Credits (Points, Score).
  - Game Window.
- Sounds.

Recognizable parts of a 2D game

What does appear in a game? Let us think about it...

- Screen.
  - Frame or Status Bar.
    - Credits (Points, Score).
    - Status.
  - Game Window.
- Sounds.

Recognizable parts of a 2D game

What does appear in a game? Let us think about it...

- Screen.
  - Frame or Status Bar.
    - Credits (Points, Score).
    - Status.
    - Chat Screen, Maps, etc.
  - Game Window.
- Sounds.

Recognizable parts of a 2D game

What does appear in a game? Let us think about it...

- Screen.
  - Frame or Status Bar.
    - Credits (Points, Score).
    - Status.
    - Chat Screen, Maps, etc.
  - Game Window.
    - Dynamic Objects (Items).
- Sounds.

Recognizable parts of a 2D game

What does appear in a game? Let us think about it...

- Screen.
  - Frame or Status Bar.
    - Credits (Points, Score).
    - Status.
    - Chat Screen, Maps, etc.
  - Game Window.
    - Dynamic Objects (Items).
    - Cursor.
- Sounds.

Recognizable parts of a 2D game

What does appear in a game? Let us think about it...

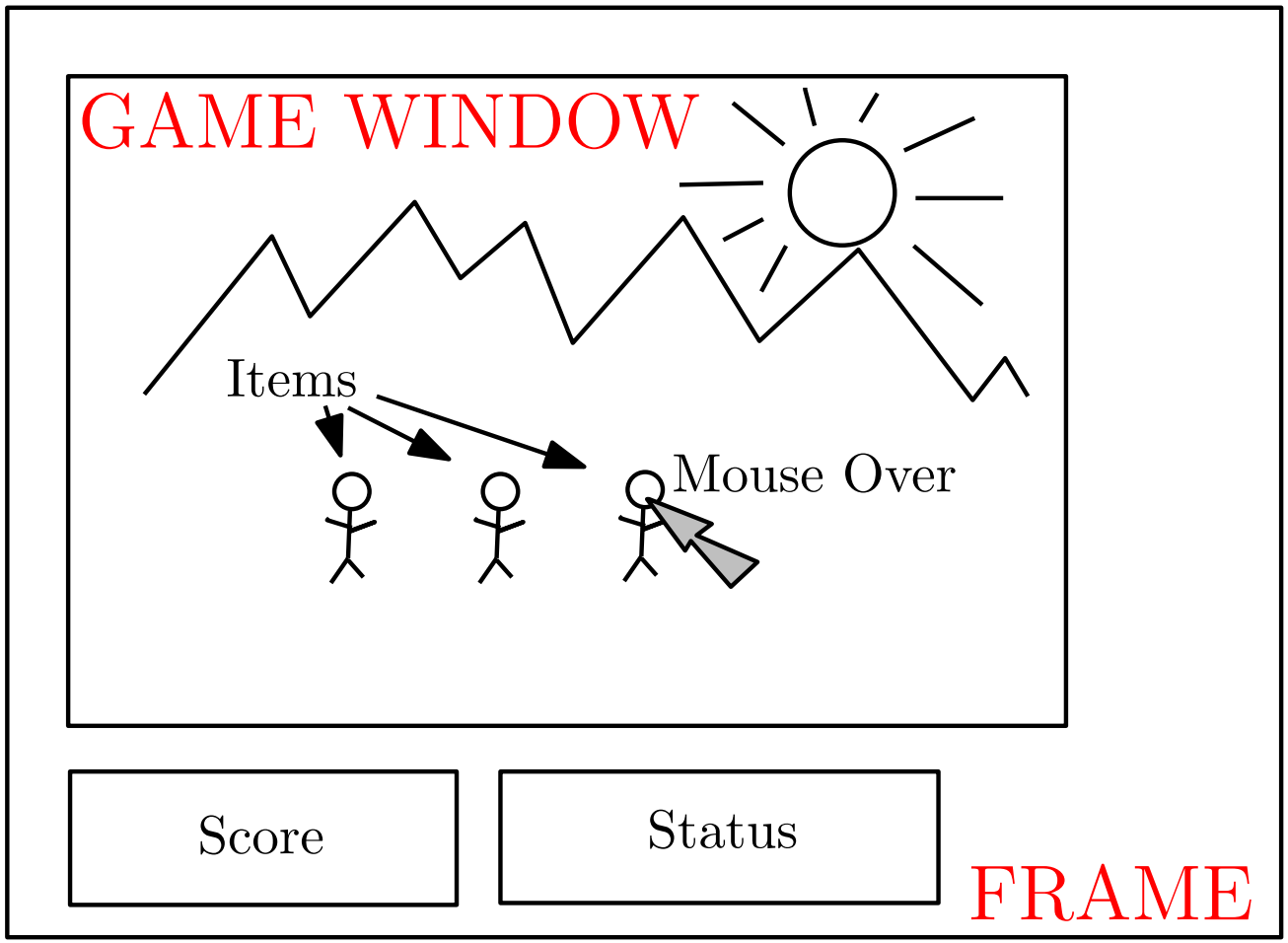
- Screen.
  - Frame or Status Bar.
    - Credits (Points, Score).
    - Status.
    - Chat Screen, Maps, etc.
  - Game Window.
    - Dynamic Objects (Items).
    - Cursor.
    - Mouse Over Texts.
- Sounds.

Recognizable parts of a 2D game

What does appear in a game? Let us think about it...

- Screen.
  - Frame or Status Bar.
    - Credits (Points, Score).
    - Status.
    - Chat Screen, Maps, etc.
  - Game Window.
    - Dynamic Objects (Items).
    - Cursor.
    - Mouse Over Texts.
    - Background.
- Sounds.

Recognizable parts of a 2D game (continued)



Transformation of the Design Analysis into a Software Model.

What do we need?

- A game class.

Transformation of the Design Analysis into a Software Model.

What do we need?

- A game class.
  - Should be something like a thread.

Transformation of the Design Analysis into a Software Model.

What do we need?

- A game class.
  - Should be something like a thread.
  - Should have access to graphic and/or audio interface.

Transformation of the Design Analysis into a Software Model.

What do we need?

- A game class.
  - Should be something like a thread.
  - Should have access to graphic and/or audio interface.
  - Should contain, control and care for all items.

Transformation of the Design Analysis into a Software Model.

What do we need?

- A game class.
  - Should be something like a thread.
  - Should have access to graphic and/or audio interface.
  - Should contain, control and care for all items.
  - Should manage or delegate user inputs and events.

Transformation of the Design Analysis into a Software Model.

What do we need?

- A game class.
  - Should be something like a thread.
  - Should have access to graphic and/or audio interface.
  - Should contain, control and care for all items.
  - Should manage or delegate user inputs and events.
  - Should be time based (game loop).

Transformation of the Design Analysis into a Software Model.

What do we need?

- A game class.
  - Should be something like a thread.
  - Should have access to graphic and/or audio interface.
  - Should contain, control and care for all items.
  - Should manage or delegate user inputs and events.
  - Should be time based (game loop).
  - Should contain the game logic.

Transformation of the Design Analysis into a Software Model.

What do we need?

- A game class.
  - Should be something like a thread.
  - Should have access to graphic and/or audio interface.
  - Should contain, control and care for all items.
  - Should manage or delegate user inputs and events.
  - Should be time based (game loop).
  - Should contain the game logic.
- A basic item class.

Transformation of the Design Analysis into a Software Model.

What do we need?

- A game class.
  - Should be something like a thread.
  - Should have access to graphic and/or audio interface.
  - Should contain, control and care for all items.
  - Should manage or delegate user inputs and events.
  - Should be time based (game loop).
  - Should contain the game logic.
- A basic item class.
  - Should have a defined status (over time).

Transformation of the Design Analysis into a Software Model.

What do we need?

- A game class.
  - Should be something like a thread.
  - Should have access to graphic and/or audio interface.
  - Should contain, control and care for all items.
  - Should manage or delegate user inputs and events.
  - Should be time based (game loop).
  - Should contain the game logic.
- A basic item class.
  - Should have a defined status (over time).
  - Should have a position.

Transformation of the Design Analysis into a Software Model.

What do we need?

- A game class.
  - Should be something like a thread.
  - Should have access to graphic and/or audio interface.
  - Should contain, control and care for all items.
  - Should manage or delegate user inputs and events.
  - Should be time based (game loop).
  - Should contain the game logic.
- A basic item class.
  - Should have a defined status (over time).
  - Should have a position.
  - Should have a dimension (area, for collision detection, etc.).

Transformation of the Design Analysis into a Software Model.

What do we need?

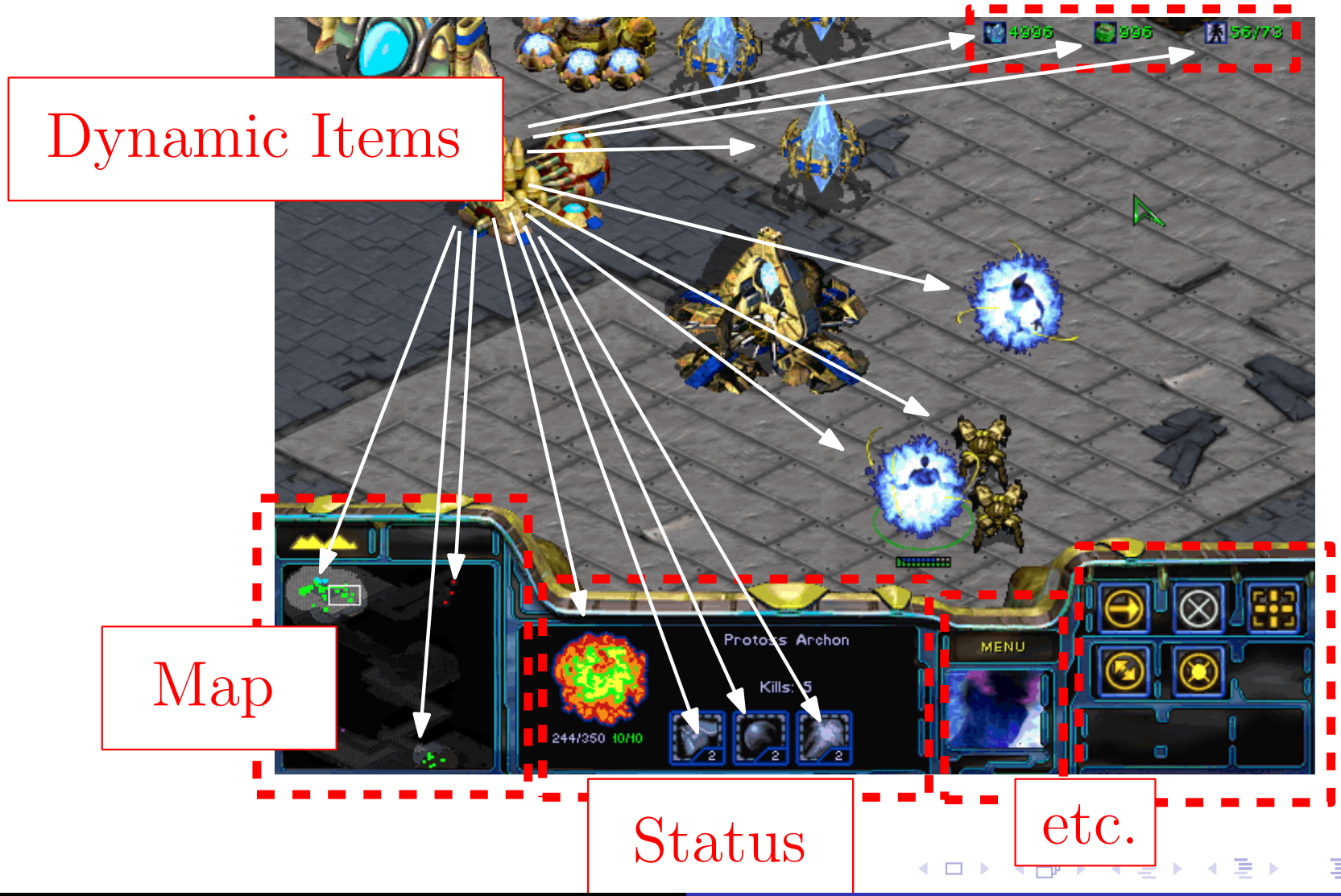
- A game class.
  - Should be something like a thread.
  - Should have access to graphic and/or audio interface.
  - Should contain, control and care for all items.
  - Should manage or delegate user inputs and events.
  - Should be time based (game loop).
  - Should contain the game logic.
- A basic item class.
  - Should have a defined status (over time).
  - Should have a position.
  - Should have a dimension (area, for collision detection, etc.).
  - Should be able to draw itself.

Transformation of the Design Analysis into a Software Model.

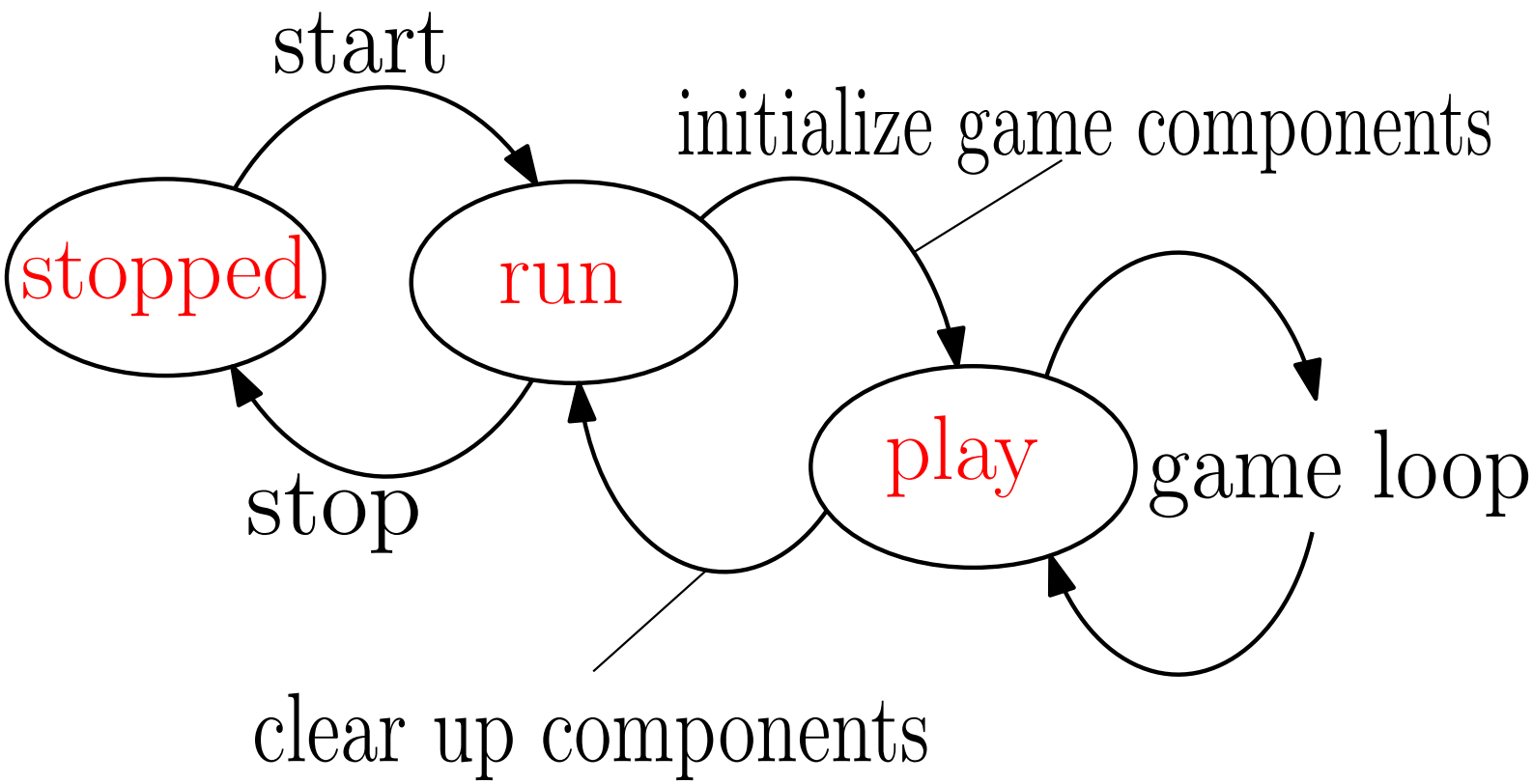
What do we need?

- A game class.
  - Should be something like a thread.
  - Should have access to graphic and/or audio interface.
  - Should contain, control and care for all items.
  - Should manage or delegate user inputs and events.
  - Should be time based (game loop).
  - Should contain the game logic.
- A basic item class.
  - Should have a defined status (over time).
  - Should have a position.
  - Should have a dimension (area, for collision detection, etc.).
  - Should be able to draw itself.
  - Should play its sounds.

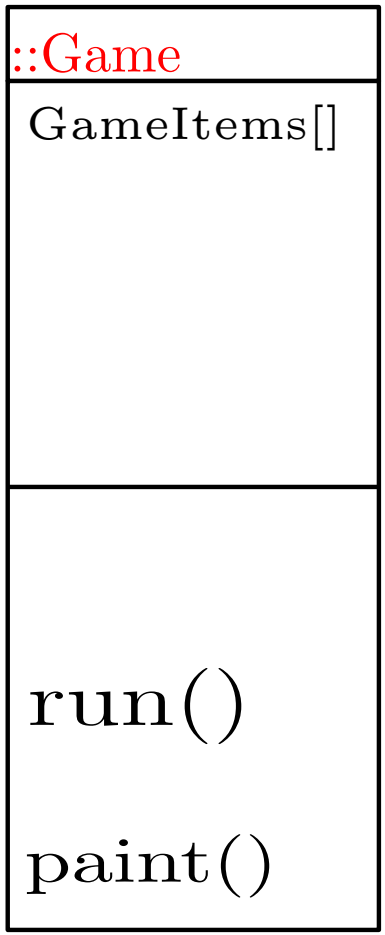
Transformation from Design Analysis to Software Model



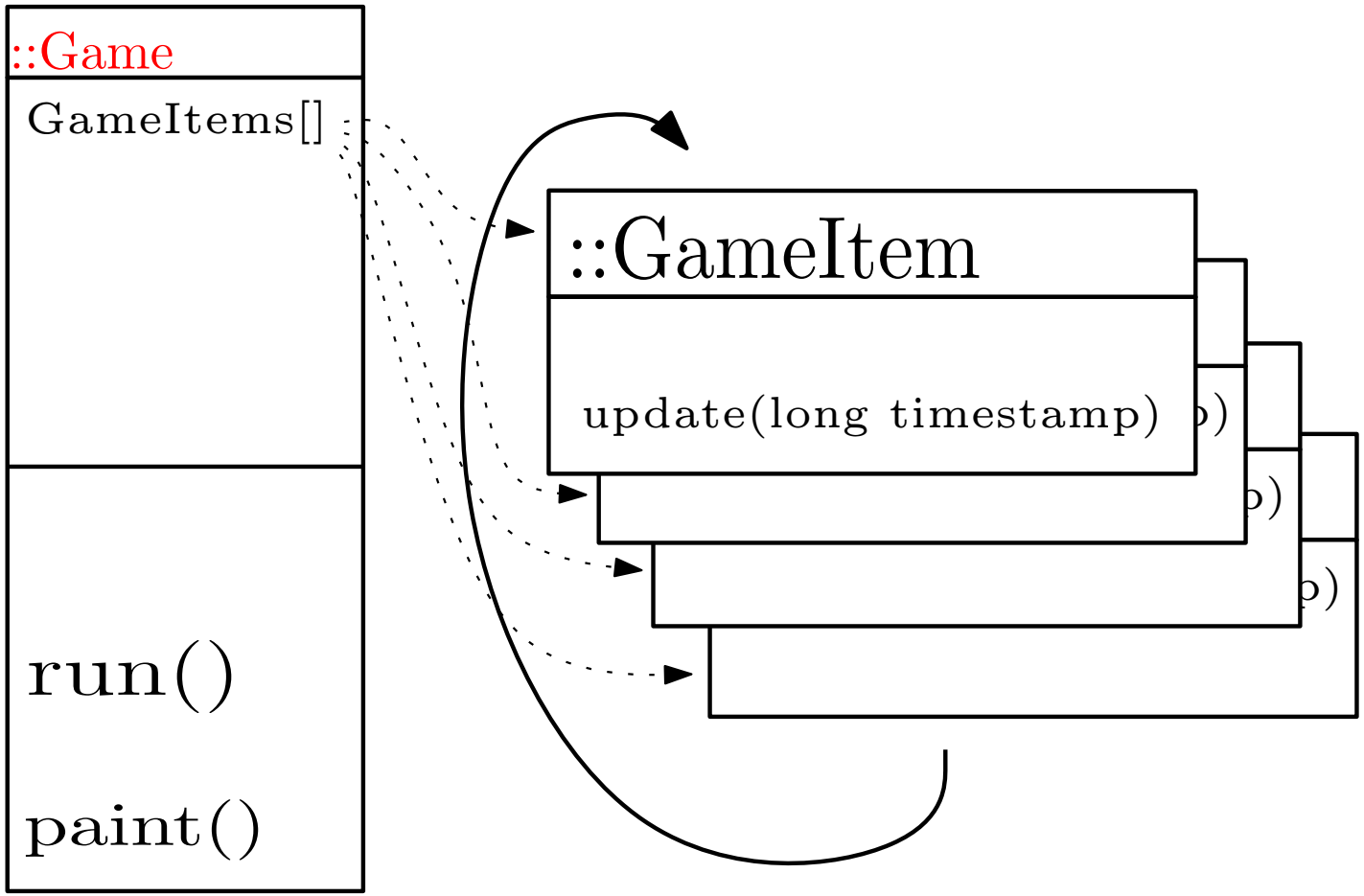
The Game class states.



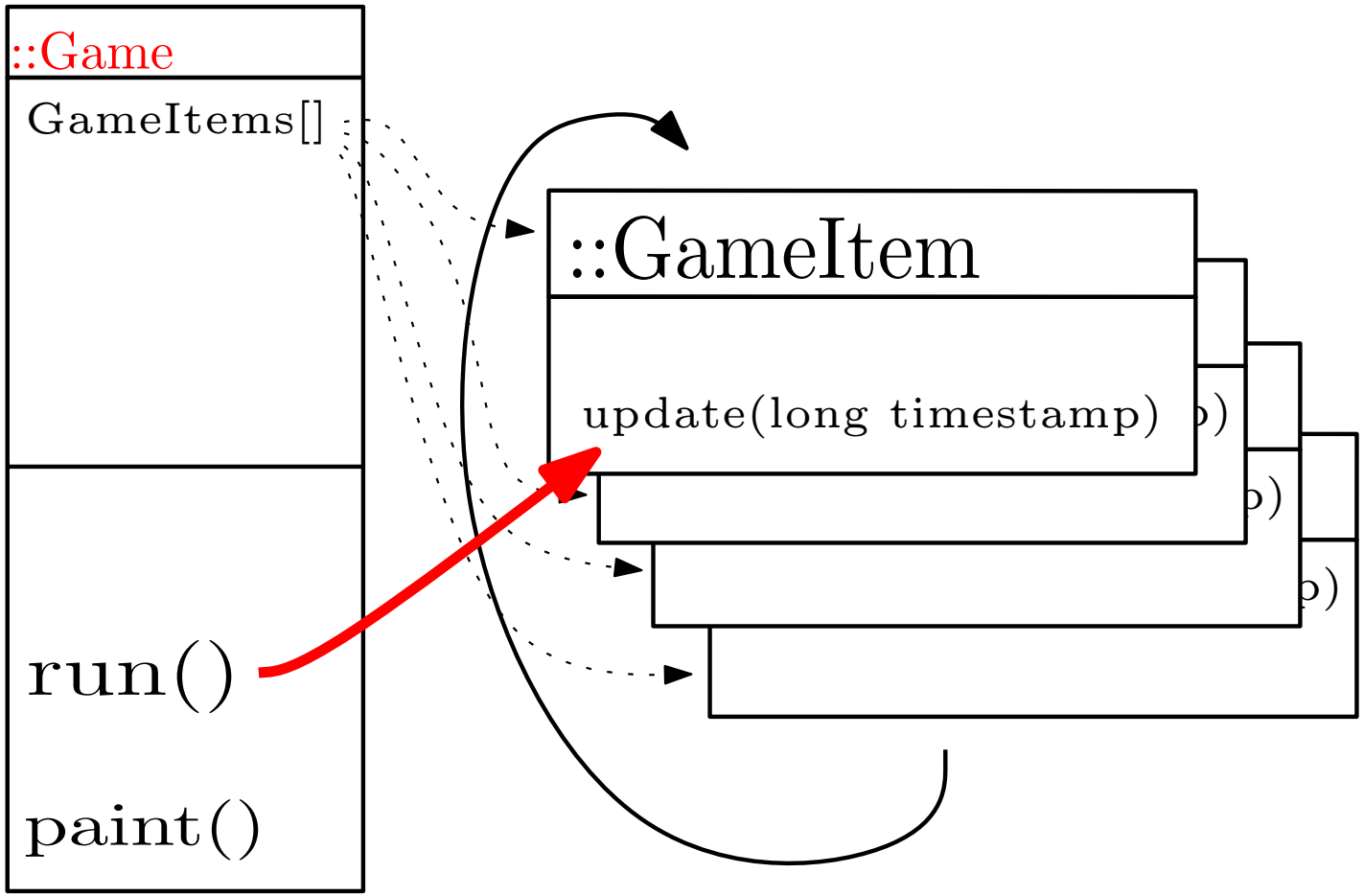
The Game Class's basic methods



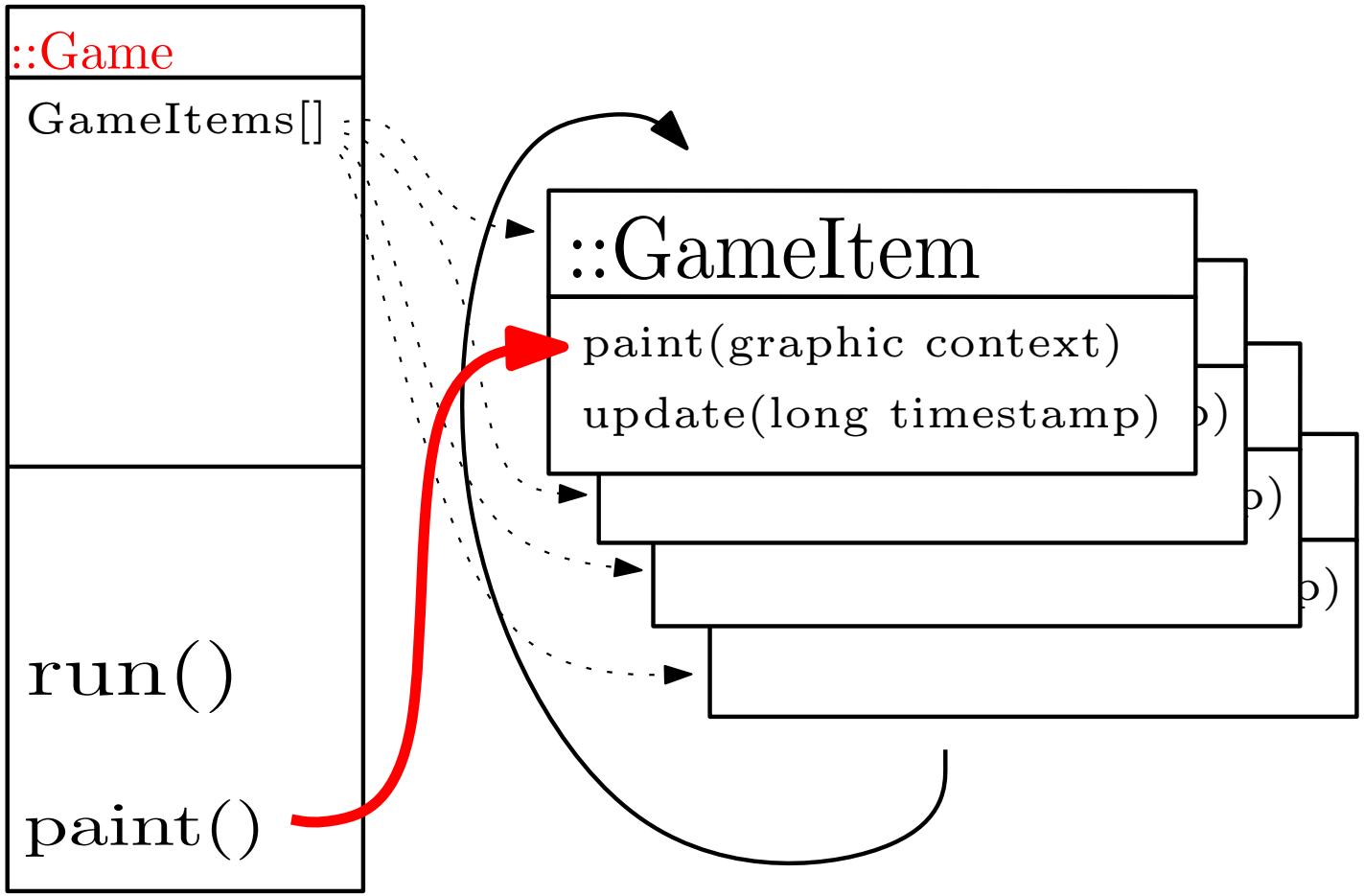
The Games' GameItem Collection



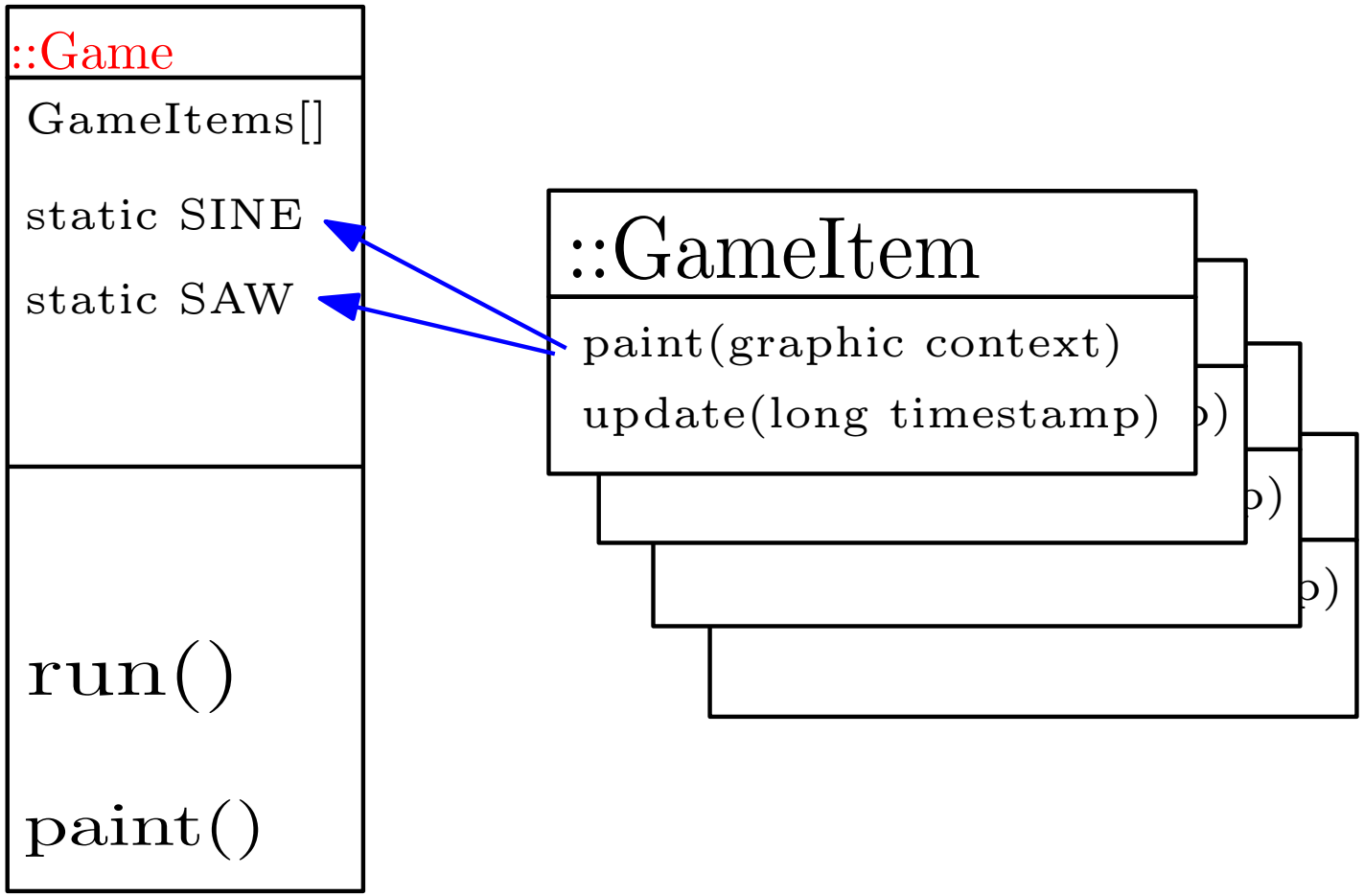
The update iteration (loop)



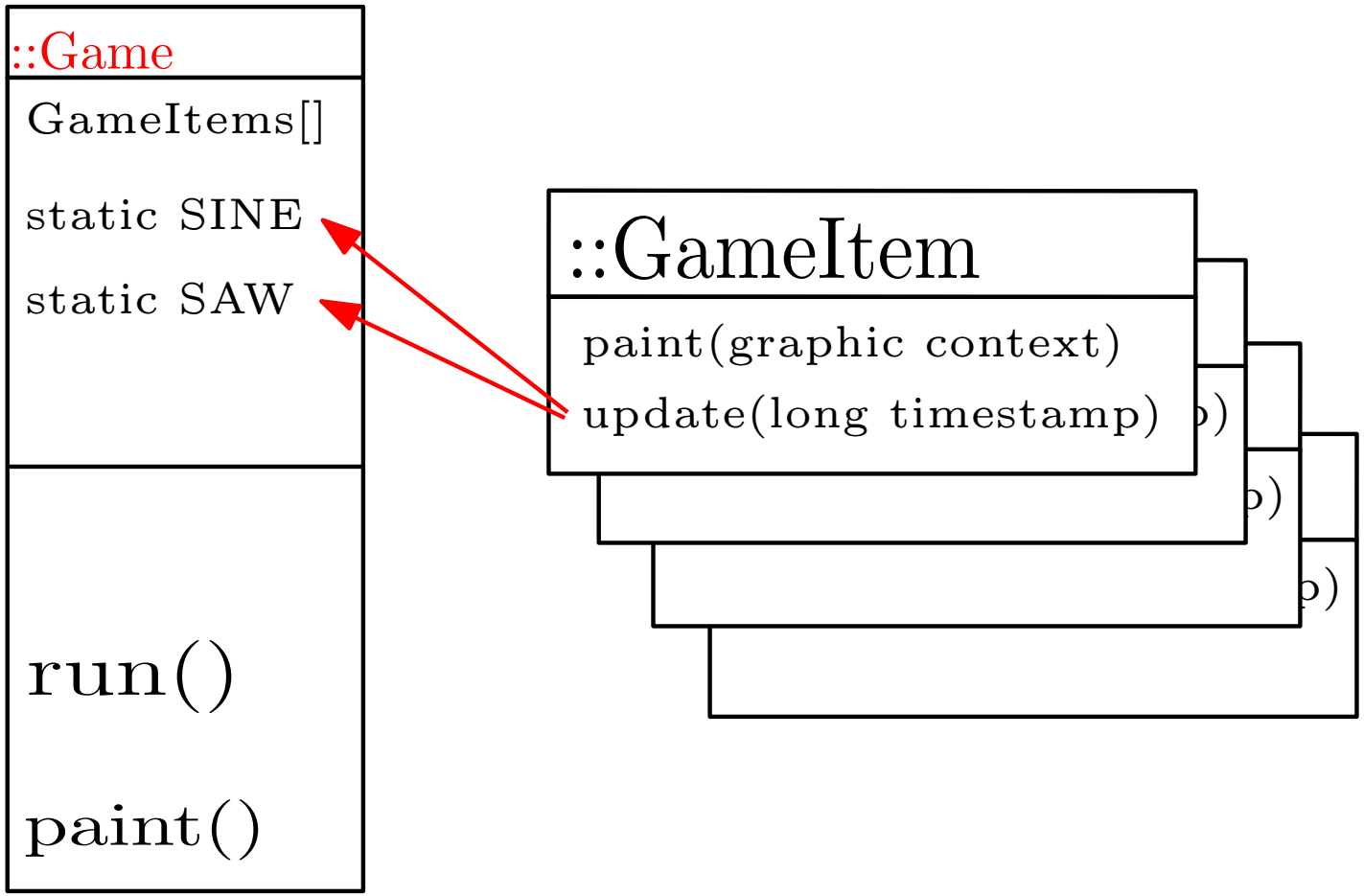
The paint iteration (loop)



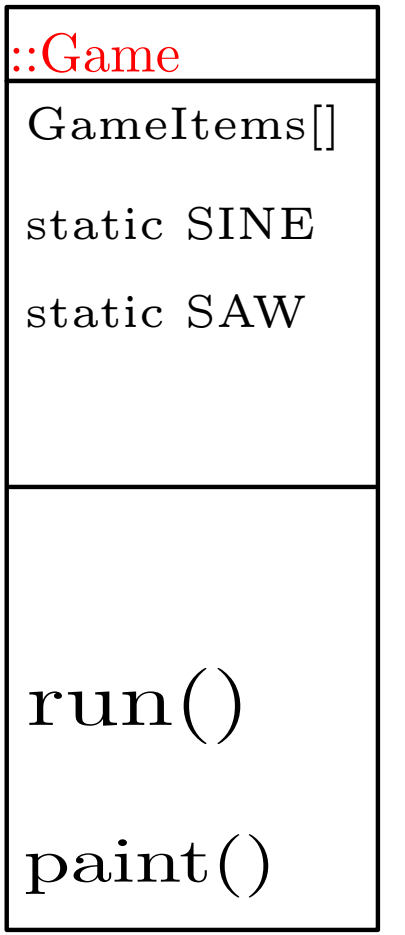
Time based global values (statics)

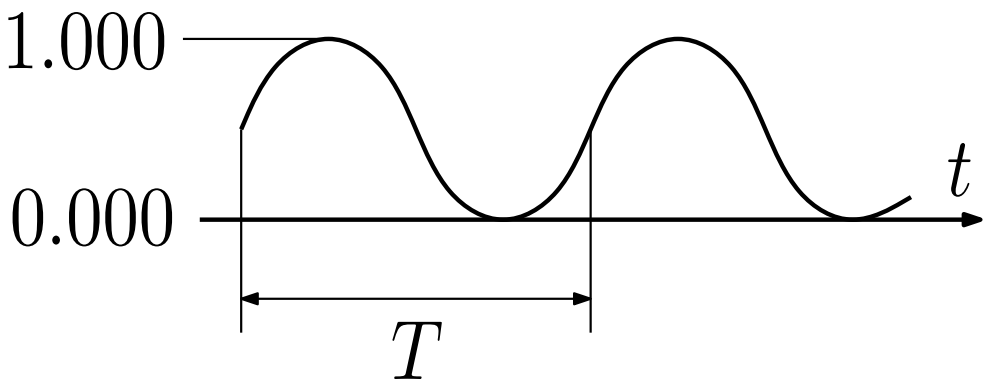


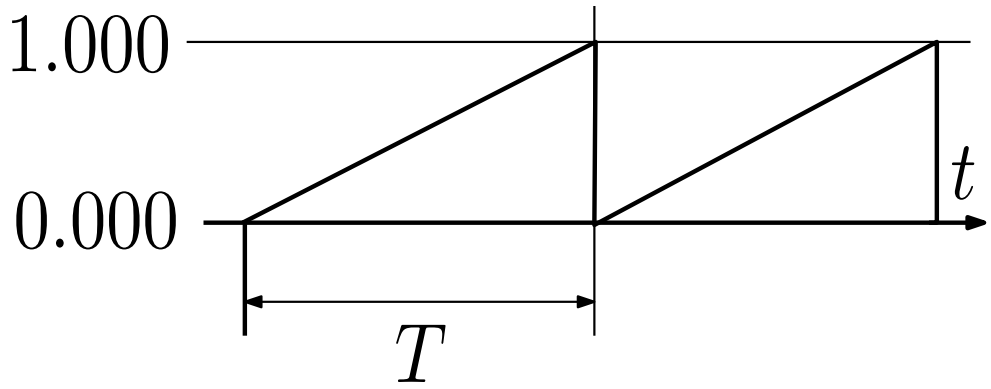
Time based global values (cont'd)



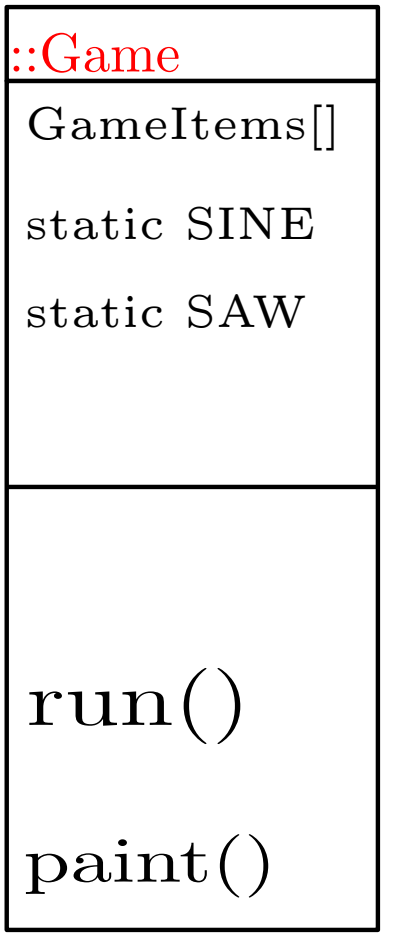
Time based global values (cont'd)

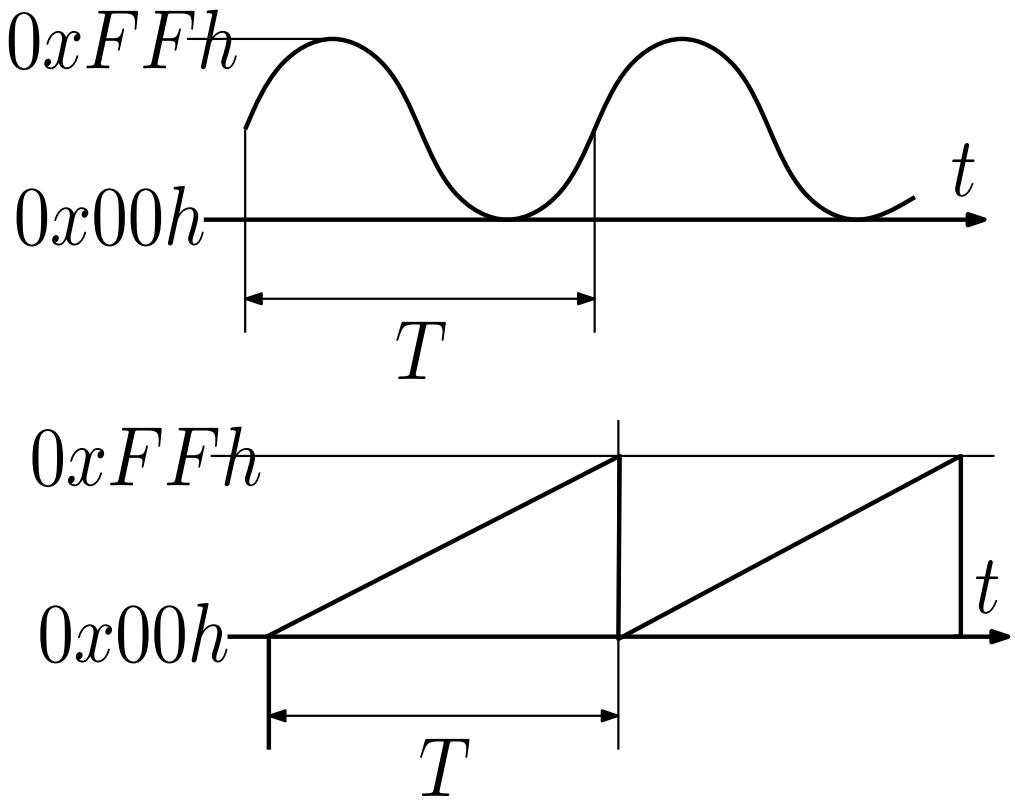




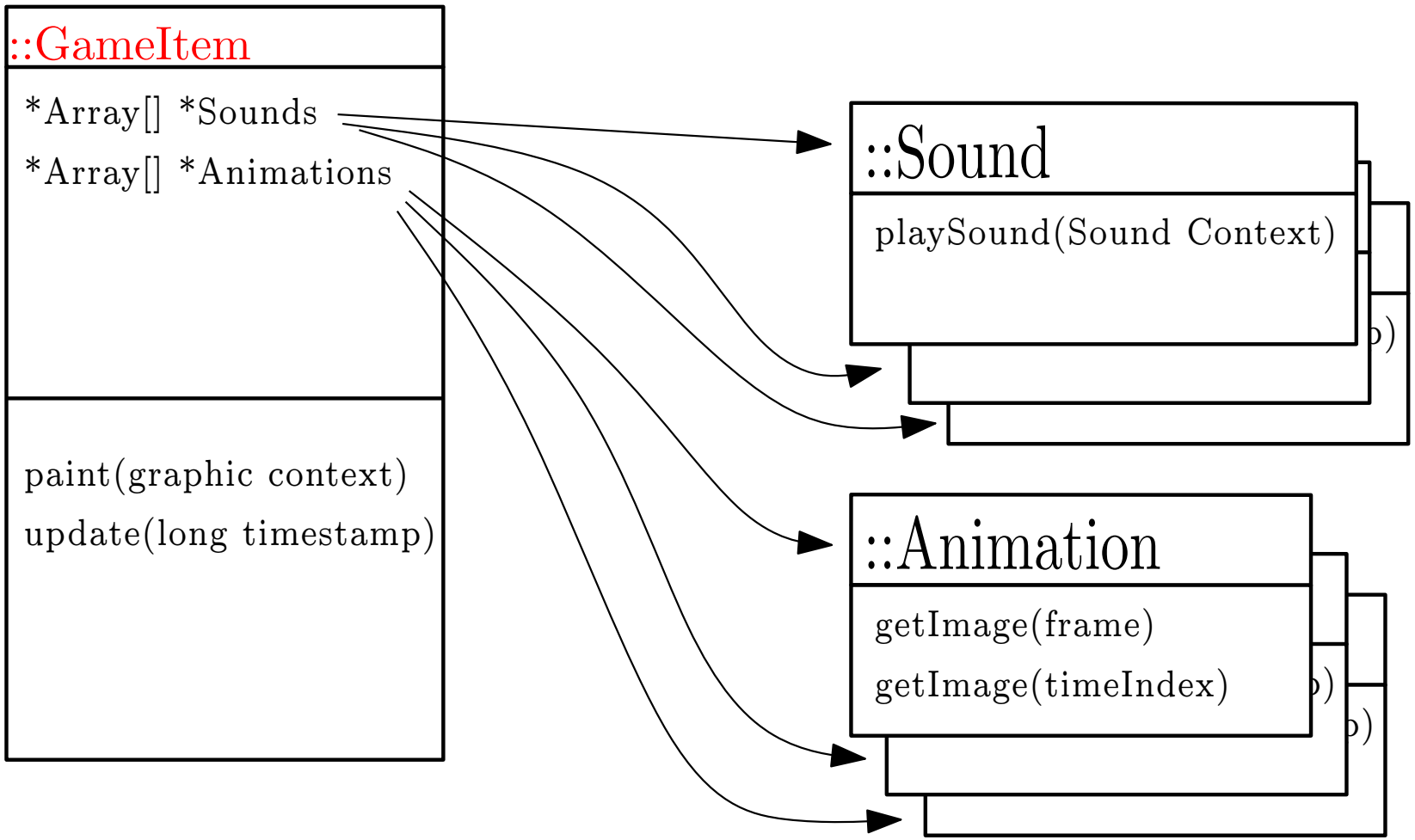


Time based global values (cont'd)

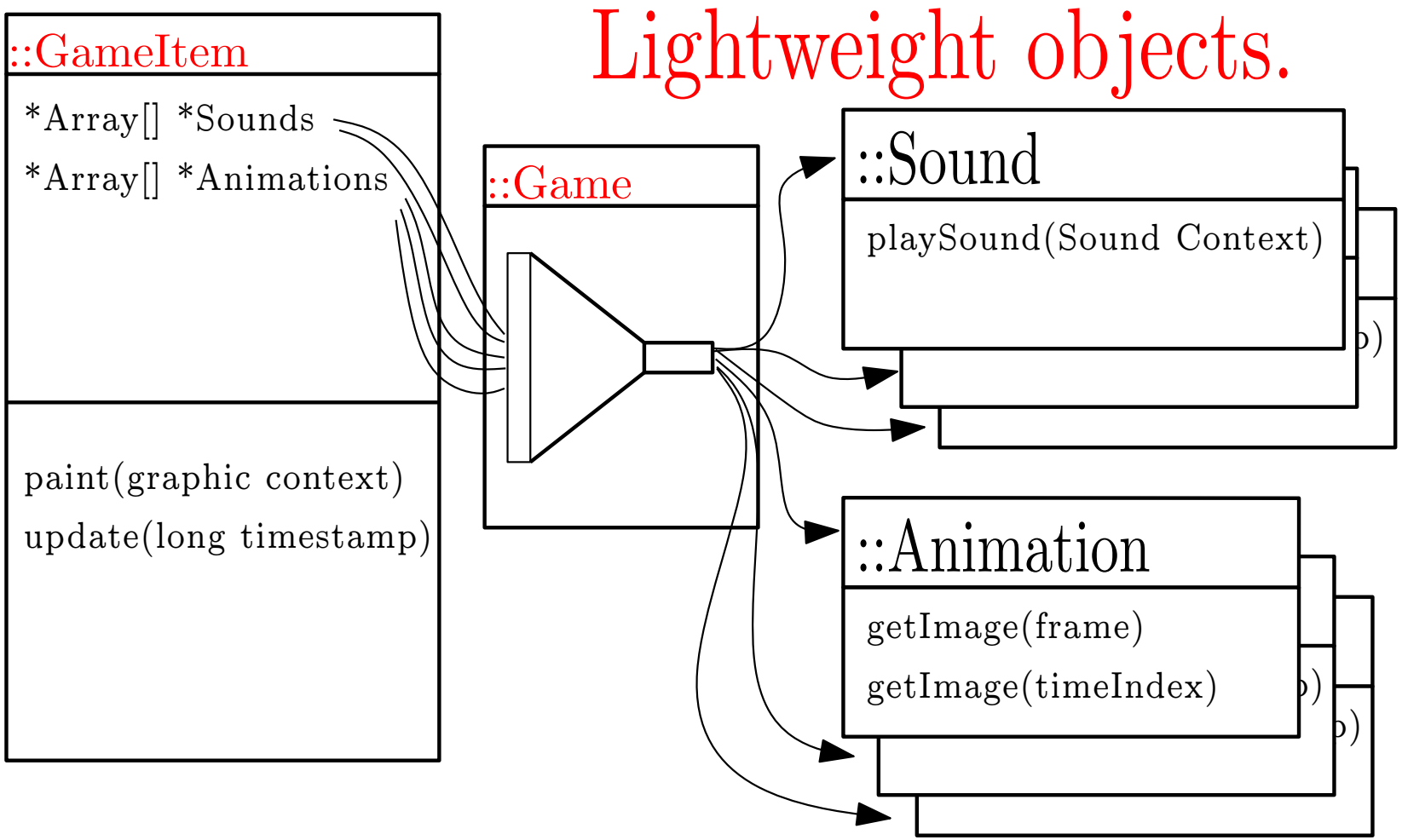




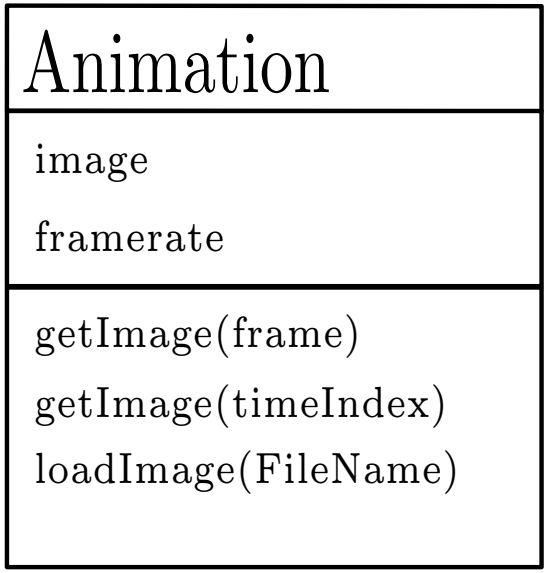
The GameItem as Container



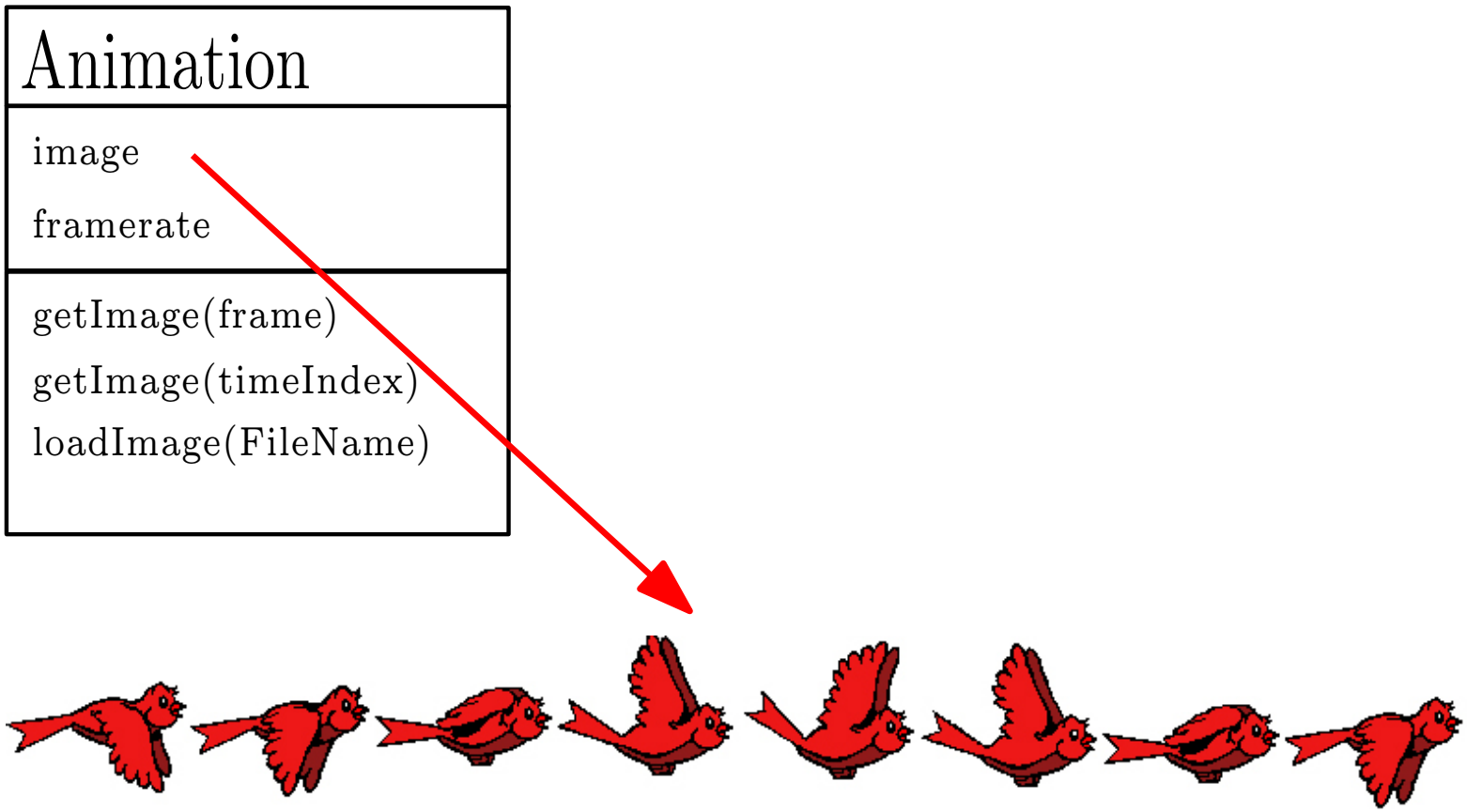
The GameItem as Container cont'd



Animation

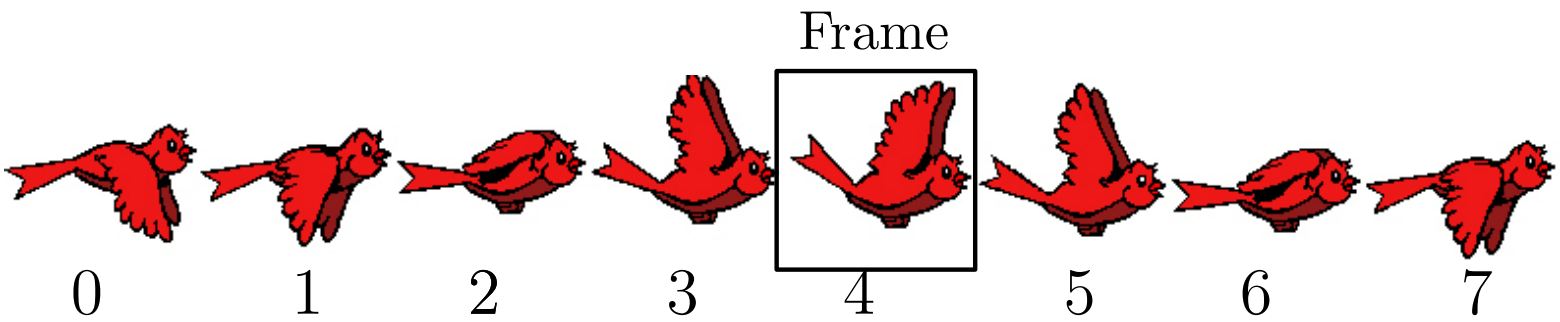

Animation
image
framerate
getImage(frame)
getImage(timeIndex)
loadImage(FileName)

Animation (cont'd)

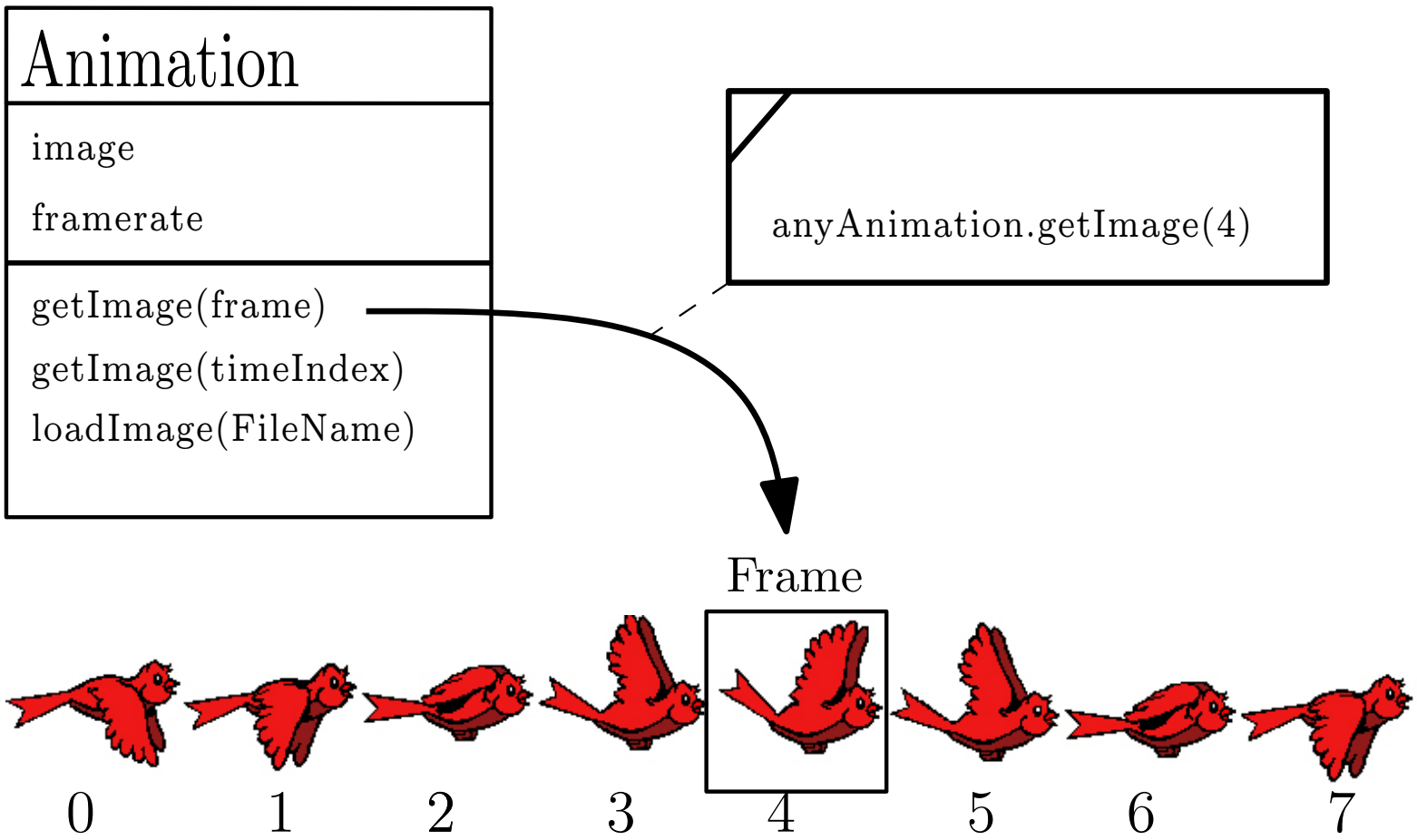


Animation (cont'd)

Animation
image framerate
getImage(frame) getImage(timeIndex) loadImage(FileName)



Animation (cont'd)



Types of Events

- The Internal Events.

Types of Events

- The Internal Events.
  - Timer Event (internal).

Types of Events

- The Internal Events.
  - Timer Event (internal).
  - GameItem Event (shoots at other Item).

Types of Events

- The Internal Events.
  - Timer Event (internal).
  - GameItem Event (shoots at other Item).
  - Collision Detection.

Types of Events

- The Internal Events.
  - Timer Event (internal).
  - GameItem Event (shoots at other Item).
  - Collision Detection.
- External Events (System Events).

Types of Events

- The Internal Events.
  - Timer Event (internal).
  - GameItem Event (shoots at other Item).
  - Collision Detection.
- External Events (System Events).
  - Events caused by User (Input Device Event).

Types of Events

- The Internal Events.
  - Timer Event (internal).
  - GameItem Event (shoots at other Item).
  - Collision Detection.
- External Events (System Events).
  - Events caused by User (Input Device Event).
  - Other System Events.

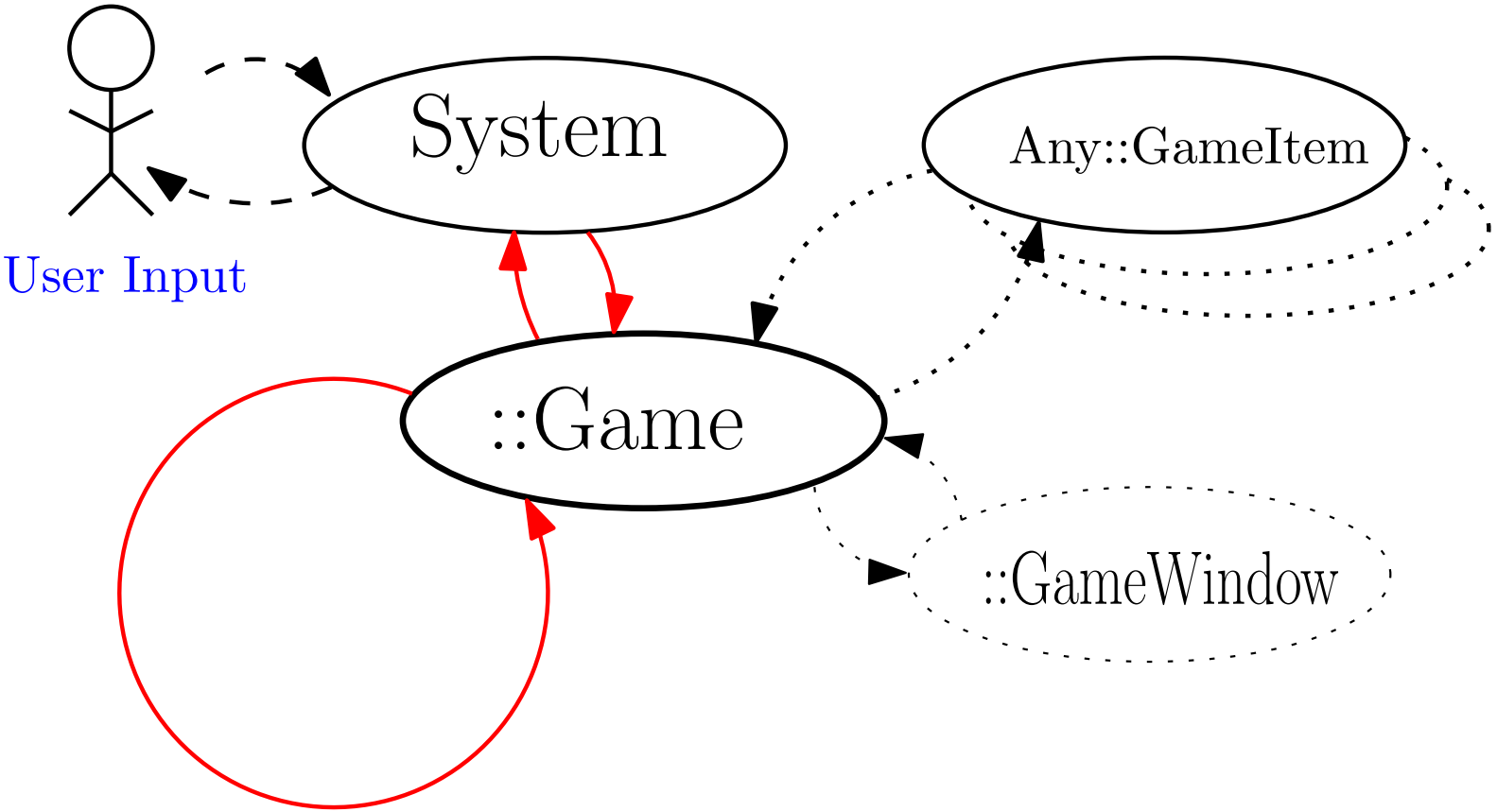
Types of Events

- The Internal Events.
  - Timer Event (internal).
  - GameItem Event (shoots at other Item).
  - Collision Detection.
- External Events (System Events).
  - Events caused by User (Input Device Event).
    - Mouse Event.
    - Keyboard Event.
    - Joystick etc.
  - Other System Events.

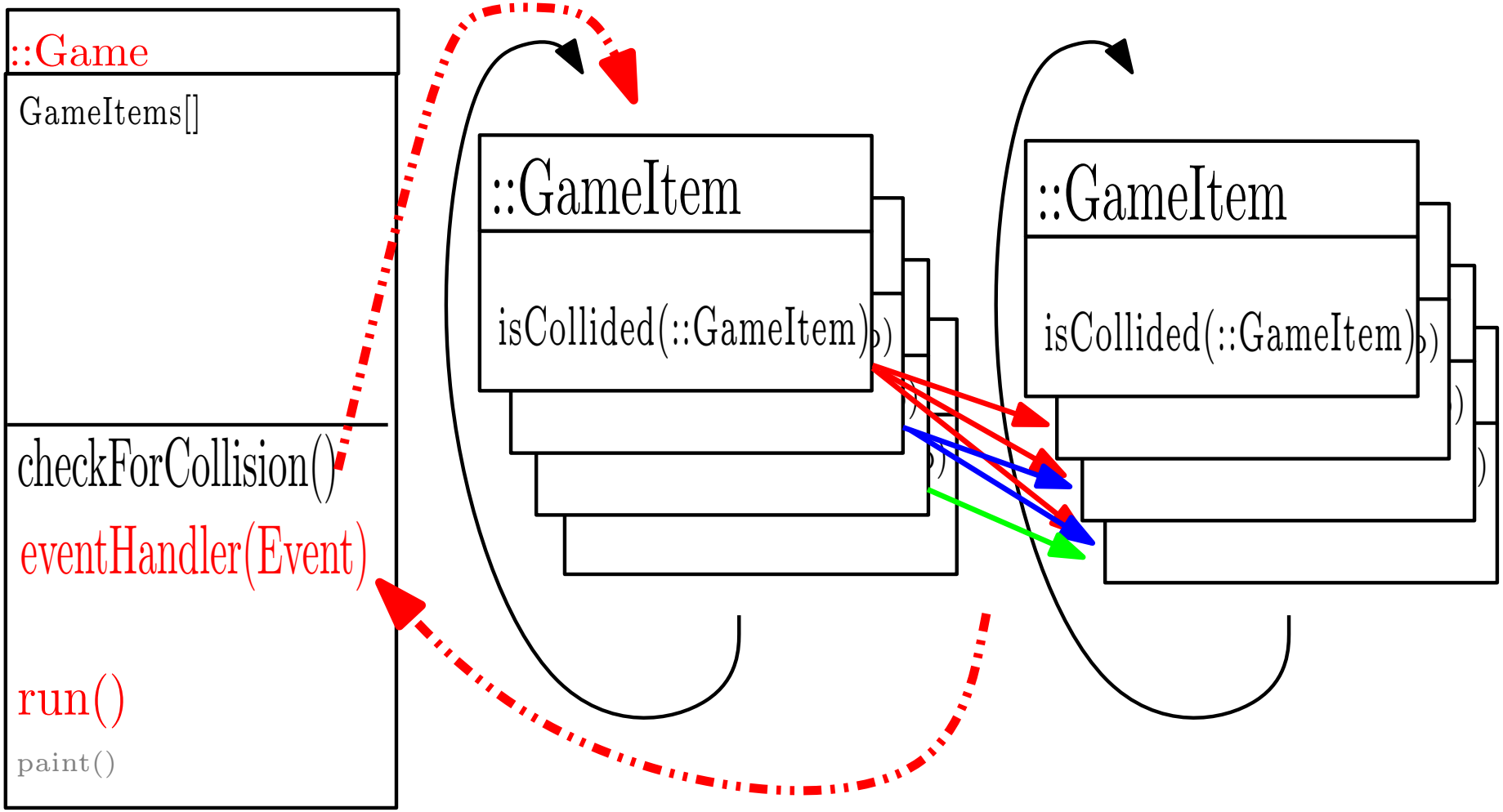
Types of Events

- The Internal Events.
  - Timer Event (internal).
  - GameItem Event (shoots at other Item).
  - Collision Detection.
- External Events (System Events).
  - Events caused by User (Input Device Event).
    - Mouse Event.
    - Keyboard Event.
    - Joystick etc.
  - Other System Events.
    - Forced Repaint.
    - Forced Shutdown.
    - Set to Background, Timer Event (external) etc.

Event processing



Collision detection



Collision detection (cont'd)

This basic algorithm would have the complexity:

$$C = \frac{n * (n-1)}{2}$$

$$O(n) = n^2$$

n... number of GameItems which may collide. Of course this can be Improved, but that is a different story...

Farewell

- Thank you for your participation...