418587: Game Design and Development
Procedural Animation

ประมุข ขันเงิน

pramook@gmail.com

ภาพเคลื่อนไหว

- เกิดจากการนำภาพนิ่งหลายๆ ภาพ มาแสดงผลต่อเนื่องกันด้วยความเร็วสูง
  - พอใช้ได้ 30 ภาพ / วินาที
  - ลื่น 60 ภาพ / วินาที

สไปรต์

- ในเกมที่มีการแสดงผลเป็นภาพสองมิติ ศิลปินจะวาดภาพนิ่งที่เรียกว่า **สไปรต์ (sprite)**
- มักจะรวมสไปรต์ที่มีความเกี่ยวเนื่องกันไว้ในไฟล์ภาพใหญ่ๆ เรียกว่า **สไปรต์ชีต (sprite sheet)**

ภาพเคลื่อนไหวในเกม

- ภาพเคลื่อนไหวที่ใช้ในเกมมีความซับซ้อน
  - มีสไปรต์หลายตัว แต่ละตัวมีการเคลื่อนไหวเป็นของตัวเอง
  - สามารถย่อขยายขนาด
  - สามารถเปลี่ยนแปลงความโปร่งใส
  - สามารถหมุนได้
  - สามารถซ้อนกันได้หลายๆ ชั้น

โครงสร้างข้อมูลสำหรับภาพเคลื่อนไหว

- เราสามารถสร้างภาพเคลื่อนไหวได้ด้วยการเขียนโปรแกรม
- แต่เขียนโปรแกรมมีข้อเสียหลายอย่าง
  - ต้องเขียนโปรแกรมใหม่สำหรับภาพเคลื่อนไหวทุกชิ้น
  - ไม่สามารถเอาภาพเคลื่อนไหวมา composite กันได้
- เราต้องการแทนภาพเคลื่อนไหวด้วย**ข้อมูล**
  - เราจะเขียน**โครงสร้างข้อมูลสำหรับเก็บภาพเคลื่อนไหว**
  - แล้วเราจะเขียนโปรแกรมเพื่อ**แปลความหมาย**ของมัน
  - แล้วจึง**แสดงผล**ภาพเคลื่อนไหวนั้น

GameLib.Animation

- โครงสร้างข้อมูลสำหรับเก็บภาพเคลื่อนไหวทั้งหมดอยู่ใน namespace GameLib.Animation
- Class ต่างๆ
  - Animation: ต้นแบบของ class สำหรับเก็บ animation ทั้งหมด
  - Renderer: ใช้วาดภาพเคลื่อนไหวทางหน้าจอ
  - TextureLoader: ใช้ load texture ที่ animation ต่างๆ ใช้
  - มี class สำหรับ animation ชนิดต่างๆ อีก 11 class

การสร้างภาพเคลื่อนไหวที่ซับซ้อน

- เราจะสร้างภาพเคลื่อนไหวที่มีความซับซ้อนได้อย่างไร?
  - เริ่มจากภาพเคลื่อนไหวง่ายๆ
    - Primitive Expressions
  - แล้วนำภาพเคลื่อนไหวง่ายๆ มาประกอบกันเป็นภาพเคลื่อนไหวที่ซับซ้อนขึ้น
    - Means of Combination
  - มองภาพเคลื่อนไหวที่ซับซ้อนเป็นภาพรวม
    - Means of Abstraction

โครงสร้างข้อมูลสำหรับภาพเคลื่อนไหว

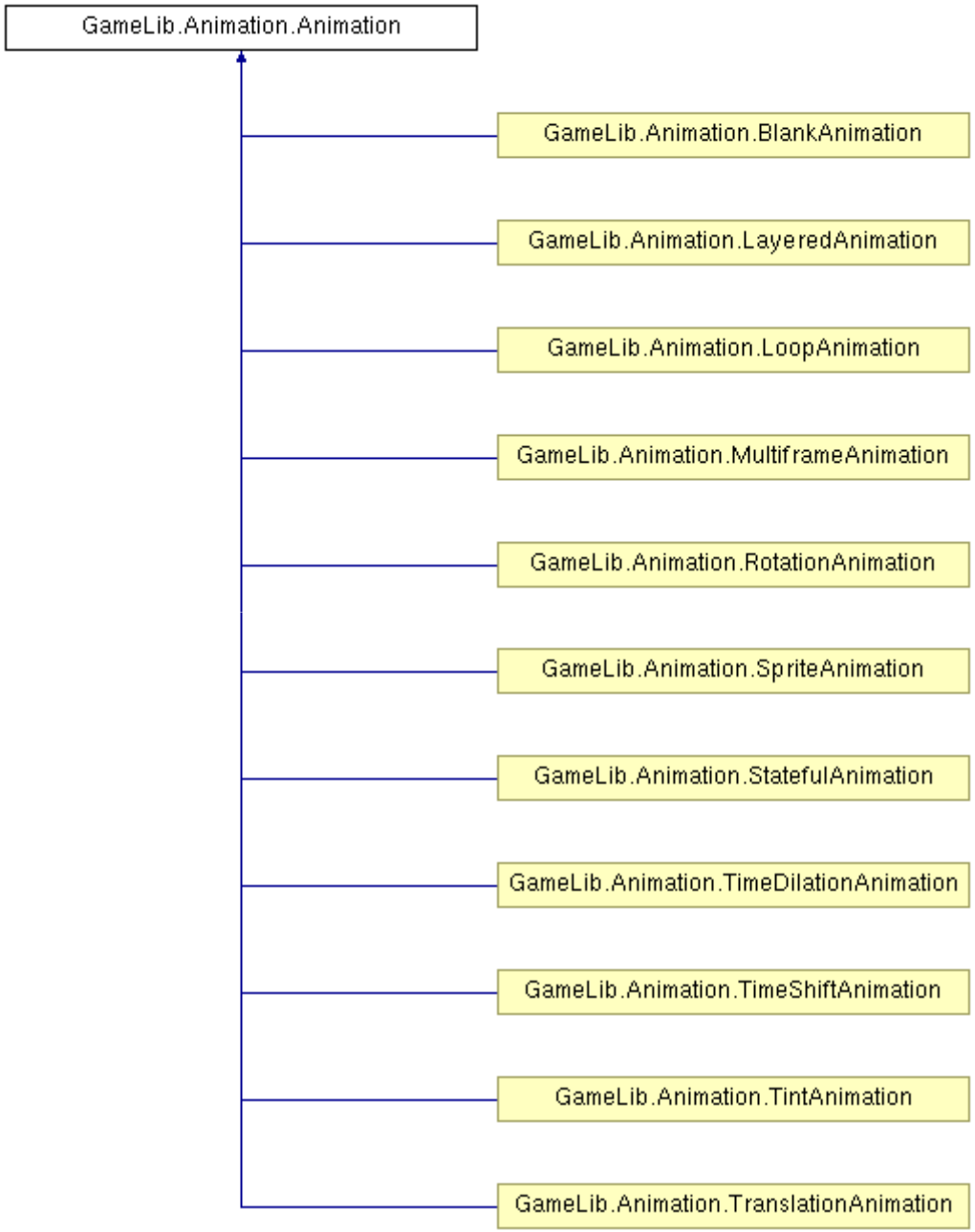
- แบ่งเป็น 3 ชนิด

1. ภาพเคลื่อนไหวพื้นฐาน
   - แบ่งแยกไม่ได้, ง่านที่สุด
2. ภาพเคลื่อนไหวเชิงซ้อน
   - เกิดจากการรวมภาพเคลื่อนไหวอื่นๆ เข้าด้วยกัน
3. ภาพเคลื่อนไหวทั่วๆ ไป
   - Class Animation
   - ภาพเคลื่อนไหวทุกอย่างมองเป็น Animation ธรรมดาเฉยๆ ได้

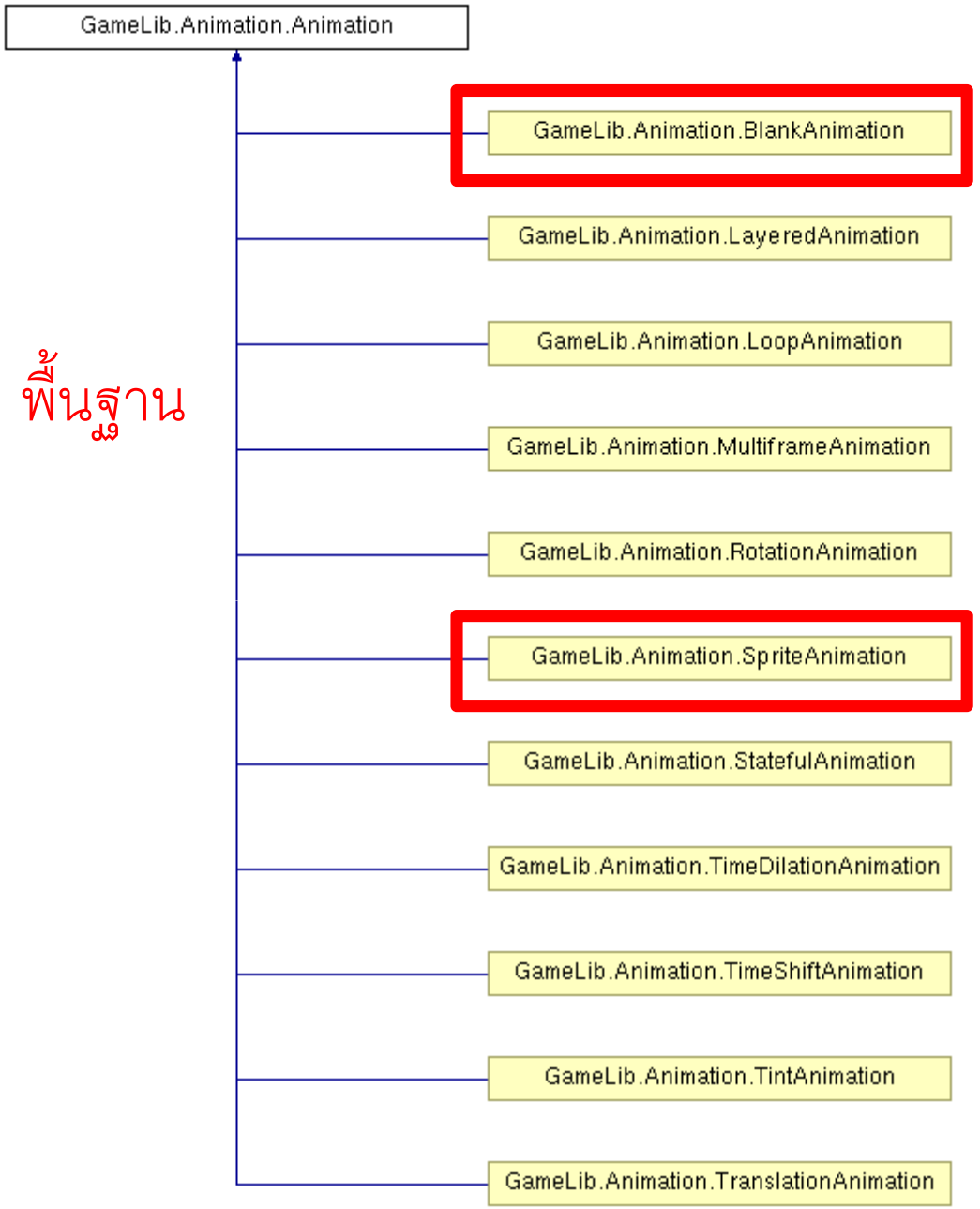
โครงสร้างข้อมูลสำหรับภาพเคลื่อนไหว

- แบ่งเป็น 3 ชนิด

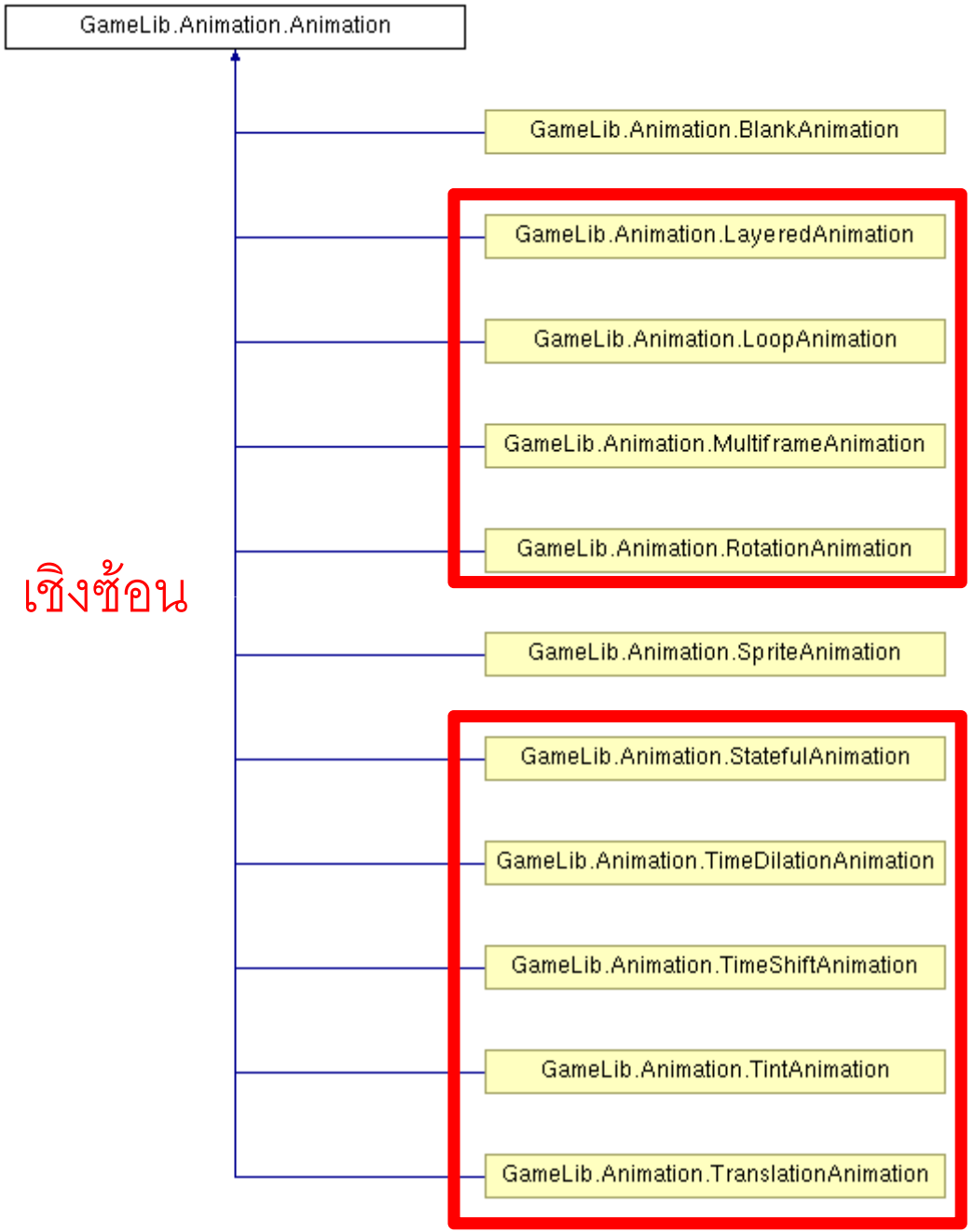
1. ภาพเคลื่อนไหวพื้นฐาน --- Primitive Expressions
   - แบ่งแยกไม่ได้, ง่านที่สุด
2. ภาพเคลื่อนไหวเชิงซ้อน --- Means of Combination
   - เกิดจากการรวมภาพเคลื่อนไหวอื่นๆ เข้าด้วยกัน
3. ภาพเคลื่อนไหวทั่วๆ ไป --- Mean of Abstraction
   - Class Animation
   - ภาพเคลื่อนไหวทุกอย่างมองเป็น Animation ธรรมดาเฉยๆ ได้


GameLib.Animation.Animation
GameLib.Animation.BlankAnimation
GameLib.Animation.LayeredAnimation
GameLib.Animation.LoopAnimation
GameLib.Animation.MultiframeAnimation
GameLib.Animation.RotationAnimation
GameLib.Animation.SpriteAnimation
GameLib.Animation.StatefulAnimation
GameLib.Animation.TimeDilationAnimation
GameLib.Animation.TimeShiftAnimation
GameLib.Animation.TintAnimation
GameLib.Animation.TranslationAnimation

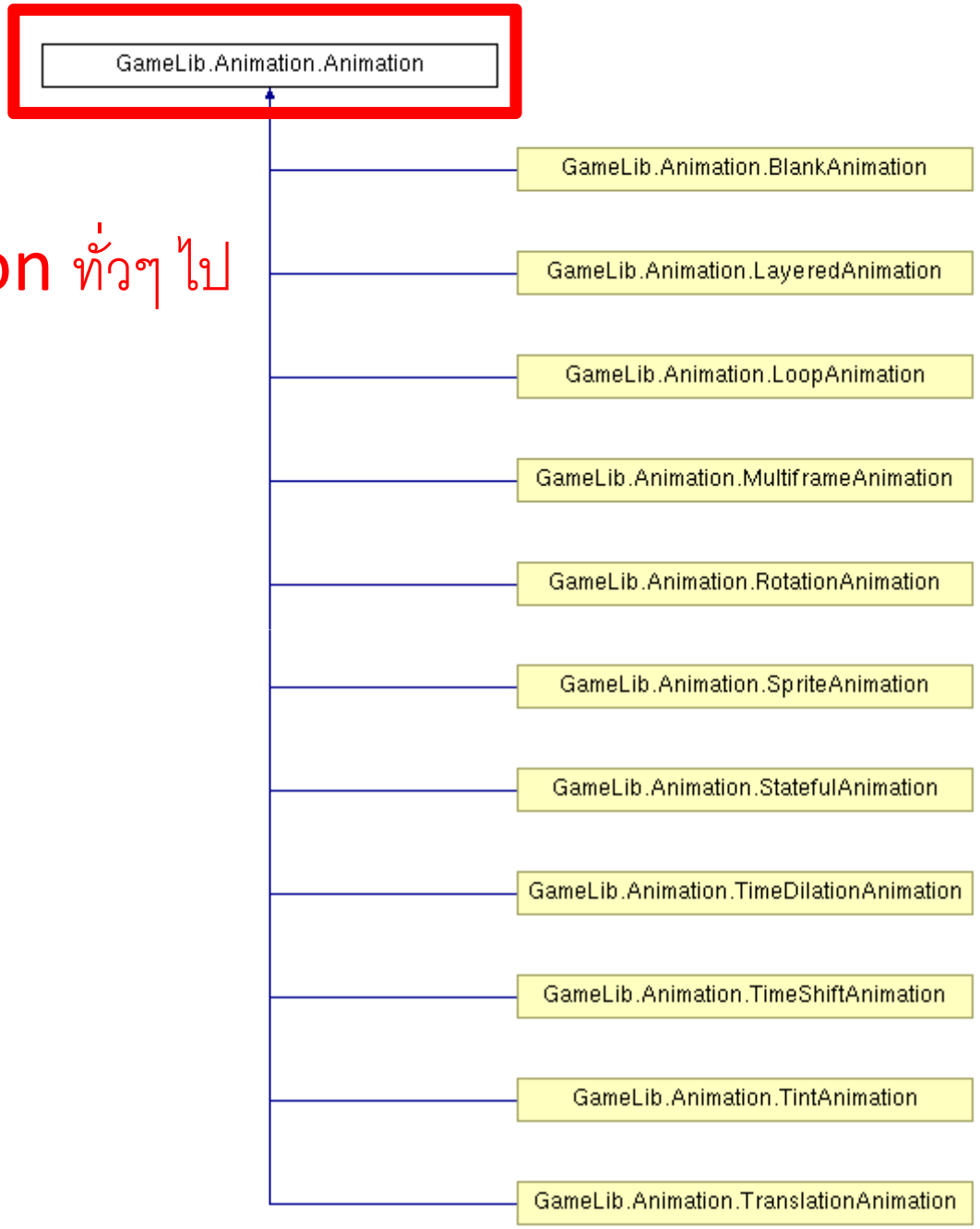
Animation พื้นฐาน



Animation เชิงซ้อน



Animation ทั่วๆ ไป



Animation แบบพื้นฐาน

- แบ่งเป็น 2 ชนิด

1. BlankAnimation
   - Animation ว่างที่ไม่แสดงรูปอะไร
   - ดูเหมือนจะไม่มีประโยชน์
   - แต่เราสามารถเอามันไปสลับกับ animation อื่น ทำให้เกิดภาพกระพริบได้
2. SpriteAnimation
   - แสดง sprite นิ่งรูปหนึ่ง

SpriteAnimation

- ข้อมูลที่ต้องเก็บใน class มีดังต่อไปนี้
  - ชื่อไฟล์ของ spritesheet
  - กรอบสี่เหลี่ยมที่บอกถึงบริเวณ spritesheet ที่จะนำไปแสดงผลในหน้าจอ
  - **ฮ็อตสป็อต (hot spot) =** จุดศูนย์กลางของภาพเคลื่อนไหว
  - ข้อมูลว่าจะต้องทำการกลับ spritesheet ก่อนนำไปแสดงผลหรือไม่

SpriteAnimation

- สมมติว่าเราจะนำ sprite จาก marisa0.png มาแสดงผล

- ต้องการภาพที่ตัวละครยืนอยู่เฉยๆ ตรงมุมบนซ้าย

SpriteAnimation

- สร้าง SpriteAnimation ด้วยคำสั่ง

```
var standAnimation = new SpriteAnimation(
    'marisa0.png',
    new Rectangle(0,0,64,64),
    new Vector2(32,63),
    SpriteEffect.None);
```

SpriteAnimation

- ผลลัพธ์คือเราได้ sprite animation ที่แสดงผลเฉพาะส่วนที่ล้อมด้วยกรอบสีเขียว
- และมี hotspot ที่จุดสีแดง

การแสดงผล

- เวลาจะนำ animation ไปแสดงเราจะเรียกเมธอด

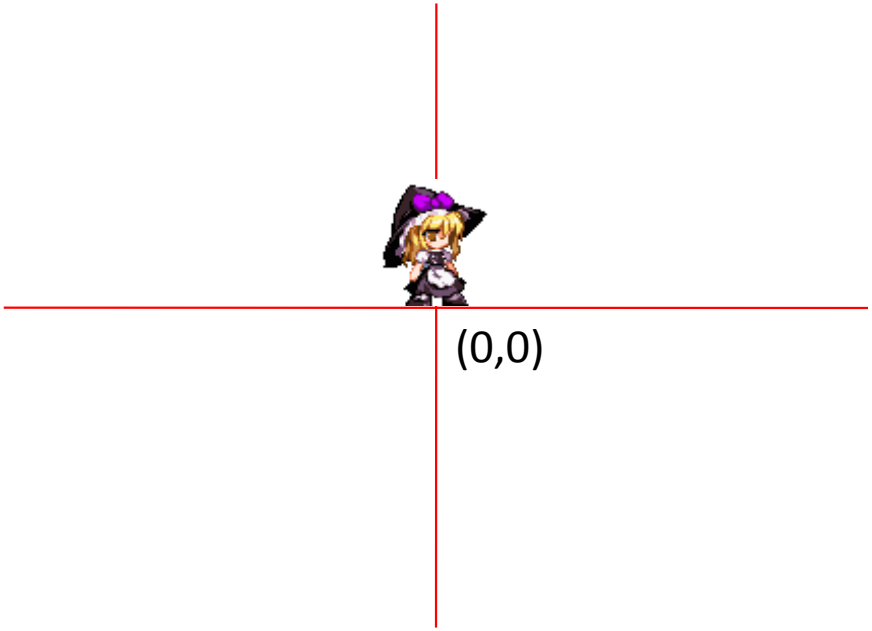
  `Animation.Render(Renderer renderer);`
- Class Renderer จะเก็บข้อมูลที่เกี่ยวข้องกับการแสดงผลเอาไว้
  - SpriteBatch
  - Matrix ของการ transform 2 มิติ (ดู 418341)
  - สีที่จะใช้วาด sprite (ที่เราต้องกำหนดให้ spriteBatch.Draw)
  - เวลา

SpriteAnimation

- สมมติว่าเราเรียก

  ```
  standAnimation.Render(renderer);
  ```

  และจะได้ผล (hotspot ไปอยู่ที่จุด (0,0))



Animation แบบเชิงซ้อน

- จะต้องมี animation อื่นๆ เป็น “ลูก”
- Animation เชิงซ้อนทำให้เกิด “กราฟ” ของ animation ต่างๆ
- เมื่อสั่งให้แสดงผล animation เชิงซ้อนหนึ่งๆ

  ลูกๆ ของมันจะถูกนำมาแสดงผลตามพฤติกรรมของ animation เชิงซ้อนนั้นๆ

Animation เชิงซ้อน

แบ่งเป็นสามพวก

1. ใช้สำหรับกำหนดลักษณะการวาดภาพ
   - TranslationAnimation
   - RotationAnimation
   - TintAnimation
   - LayeredAnimation

Animation เชิงซ้อน

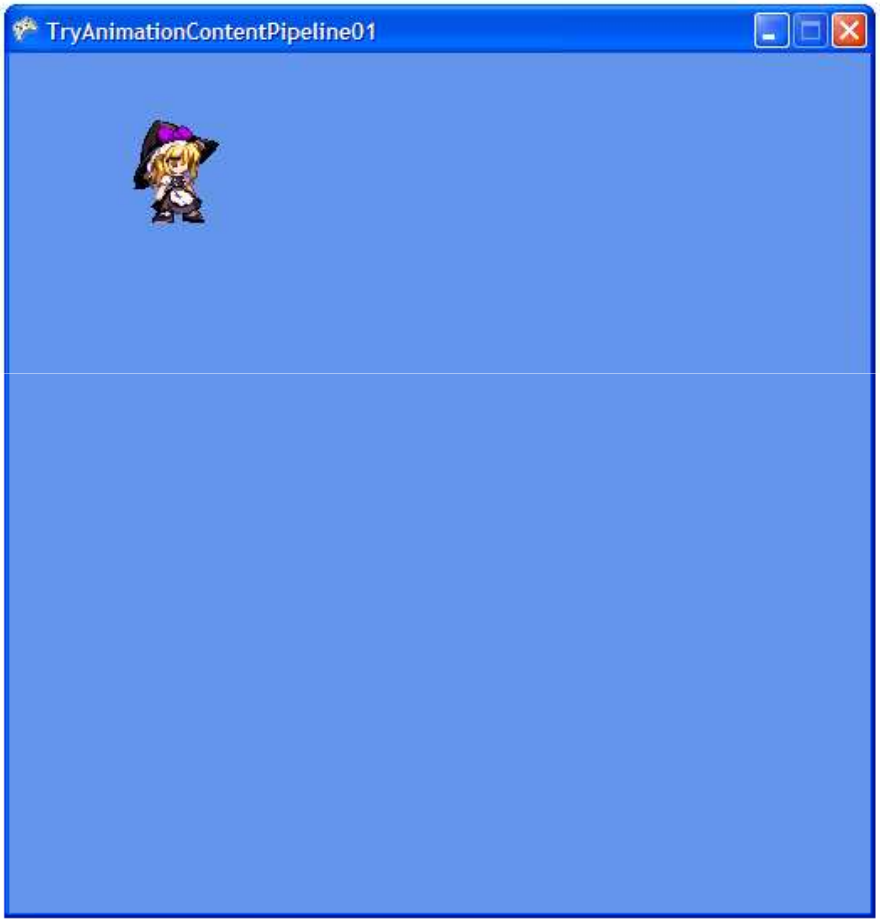
2. ใช้สำหรับกำหนดการแสดงผลตามเวลา
   - TimeShiftAnimation
   - TimeDilationAnimation
   - MultiframeAnimation
   - LoopAnimation
3. สำหรับเลือก animation มาแสดงผลตาม “สถานะ”
   - StatefulAnimation

TranslationAnimation

- ใช้กำหนดตำแหน่ง (0,0) ใหม่
- สมมติว่าเราต้องการแสดงผล “stand” ใหม่โดยให้จุด hotspot ไปอยู่ที่จุด (100, 100)
- ให้สร้าง

```
animation1 =
    new TranslationAnimation(
        stand, ← ลูก
        new Vector2(100,100)); ← การขจัด
```

TranslationAnimation

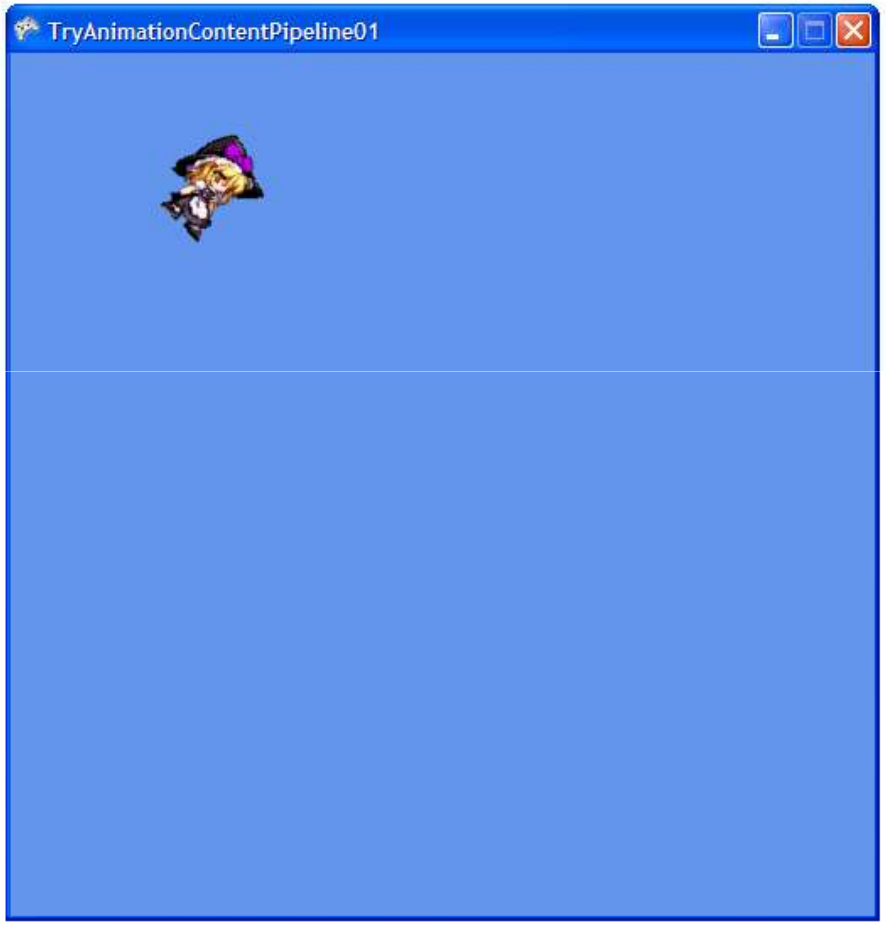


RotationAnimation

- ใช้สำหรับการหมุน sprite รอบ hotspot ของมัน

```
animation2 = new TranslationAnimation(
    new RotationAnimation(
        standAnimation, ← ลูก
        Math.PI / 4), ← มุมที่หมุนมีหน่วยเป็นเรเดียน
    new Vector2(100, 100));
```

RotationAnimation

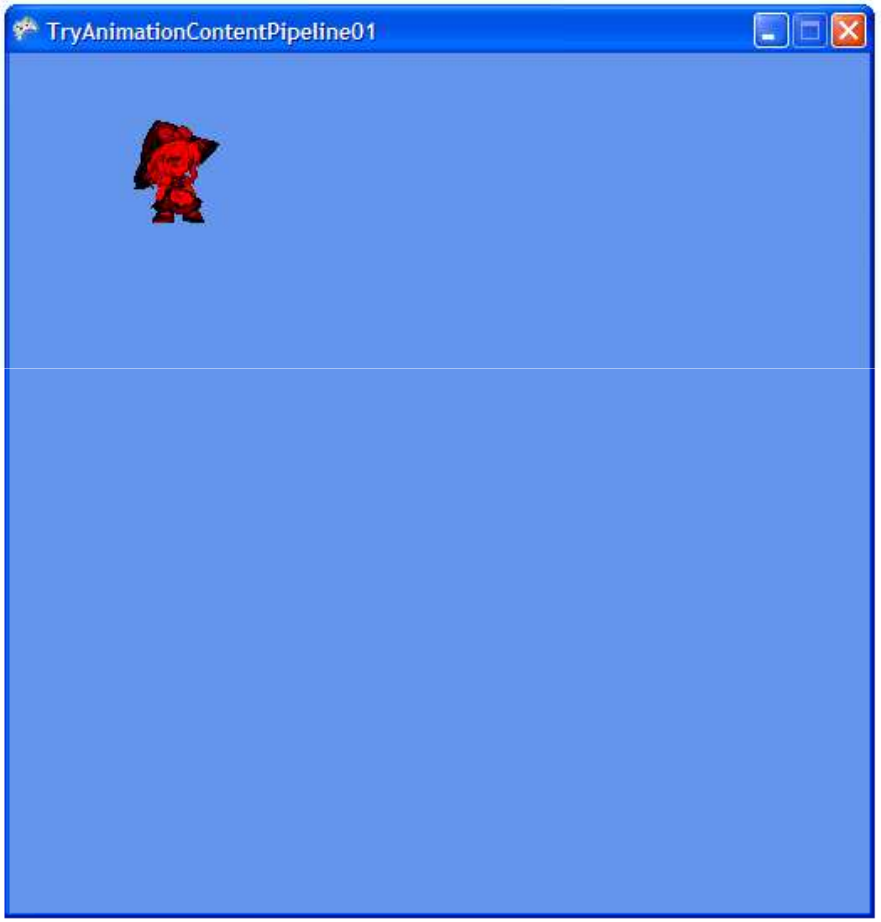


TintAnimation

- คูณสีของภาพด้วยสีที่กำหนด

```
animation3 = new TranslationAnimation(
    new TintAnimation(
        standAnimation, ← ลูก
        Color.Red), ← สี
    new Vector2(100, 100));
```

TintAnimation



LayeredAnimation

- ภาพซ้อนกันหลายๆ ชั้น

```
var walk1 = new SpriteAnimation(
  @"texture\marisa00.png",
  new Rectangle(64, 64, 64, 64),
  new Vector2(32, 63));
var walk2 = new SpriteAnimation(
  @"texture\marisa00.png",
  new Rectangle(128, 64, 64, 64),
  new Vector2(32, 63));
var walk3 = new SpriteAnimation(
  @"texture\marisa00.png",
  new Rectangle(192, 64, 64, 64),
  new Vector2(32, 63));
```

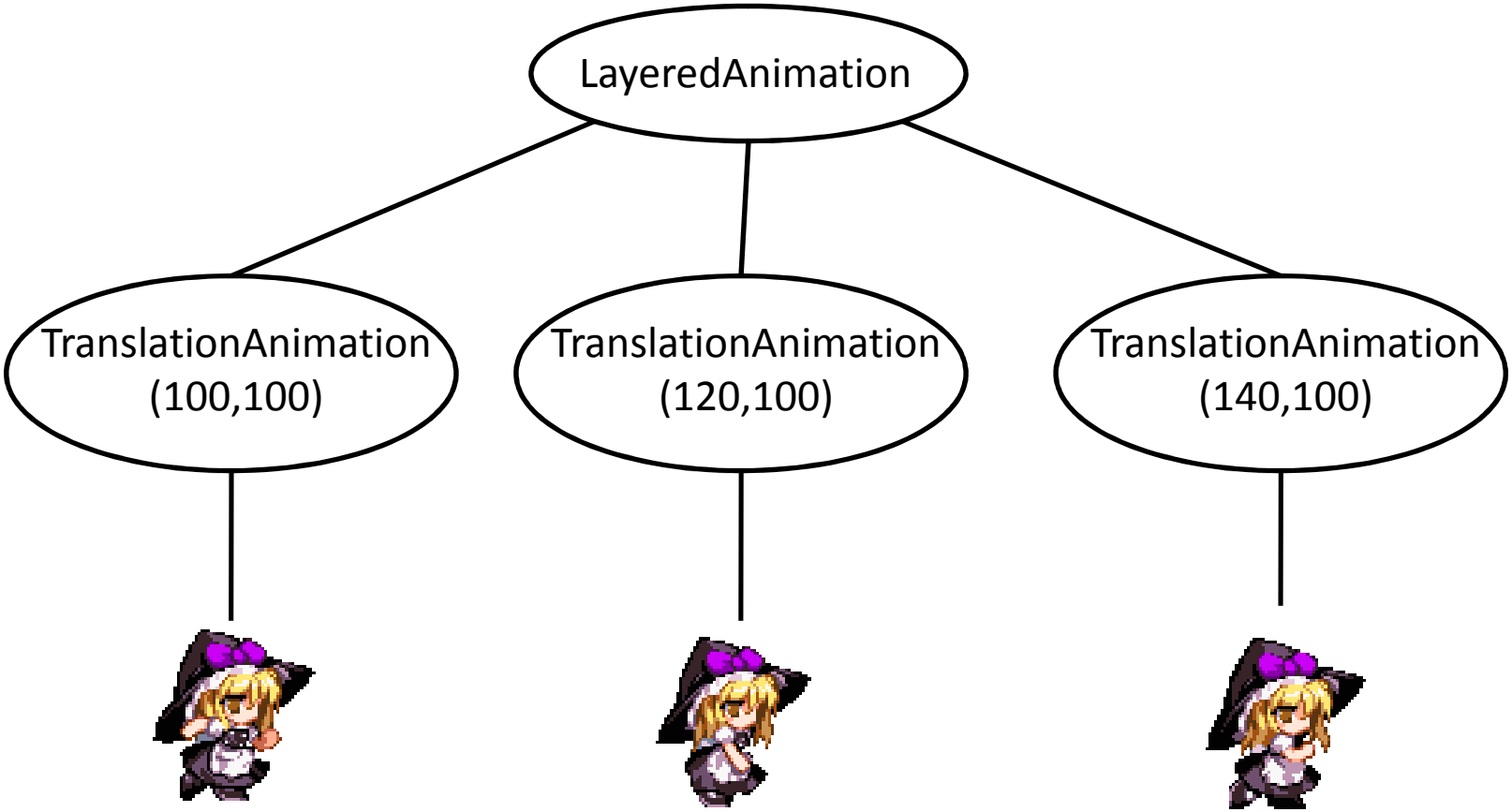
LayeredAnimation

walk1 walk2 walk3

LayeredAnimation

```
layeredAnimation = new LayeredAnimation();
layeredAnimation.AddAnimation(
  new TranslationAnimation(
    walk1,
    new Vector2(100, 100)));
layeredAnimation.AddAnimation(
  new TranslationAnimation(
    walk2,
    new Vector2(120, 100)));
layeredAnimation.AddAnimation(
  new TranslationAnimation(
    walk3,
    new Vector2(140, 100)));
```

LayeredAnimation



LayeredAnimation

MultiframeAnimation

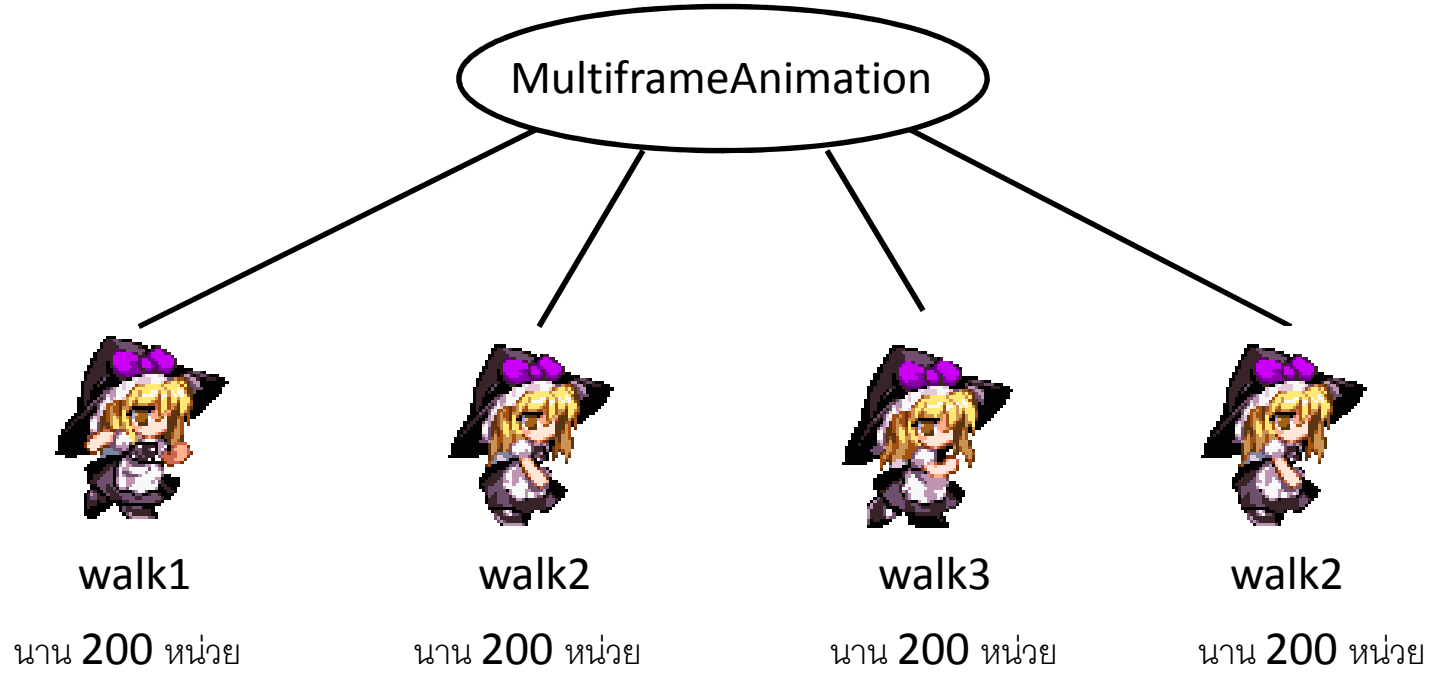
- ใช้สร้าง **animation** ที่ประกอบด้วย “เฟรม” (ภาพนิ่งหนึ่งภาพ) ถูกแสดงต่อเนื่องกันตามเวลา

```
var multiframeAnimation = new
  MultiframeAnimation();
multiframeAnimation.Add(walk1, 200);
multiframeAnimation.Add(walk2, 200);
multiframeAnimation.Add(walk3, 200);
multiframeAnimation.Add(walk2, 200);
```

← ลูกและความยาว

MultiframeAnimation

- อนิเมชันที่เราสร้างมี 4 frame



MultiframeAnimation

- เวลาแสดงผล ให้เซตเวลาของ renderer ก่อนด้วย

```
renderer.Time =
  gameTime.TotalGameTime.TotalMilliseconds;
spriteBatch.Begin();
multiframeAnimation.Render(renderer);
spriteBatch.End();
```

- การเซตเวลาของ renderer แบบนี้จะทำให้เวลามีหน่วยเป็นมิลลิวินาที
- คุณสามารถเซตให้เวลามีหน่วยเป็นอะไรก็ได้ตามต้องการ

MultiframeAnimation

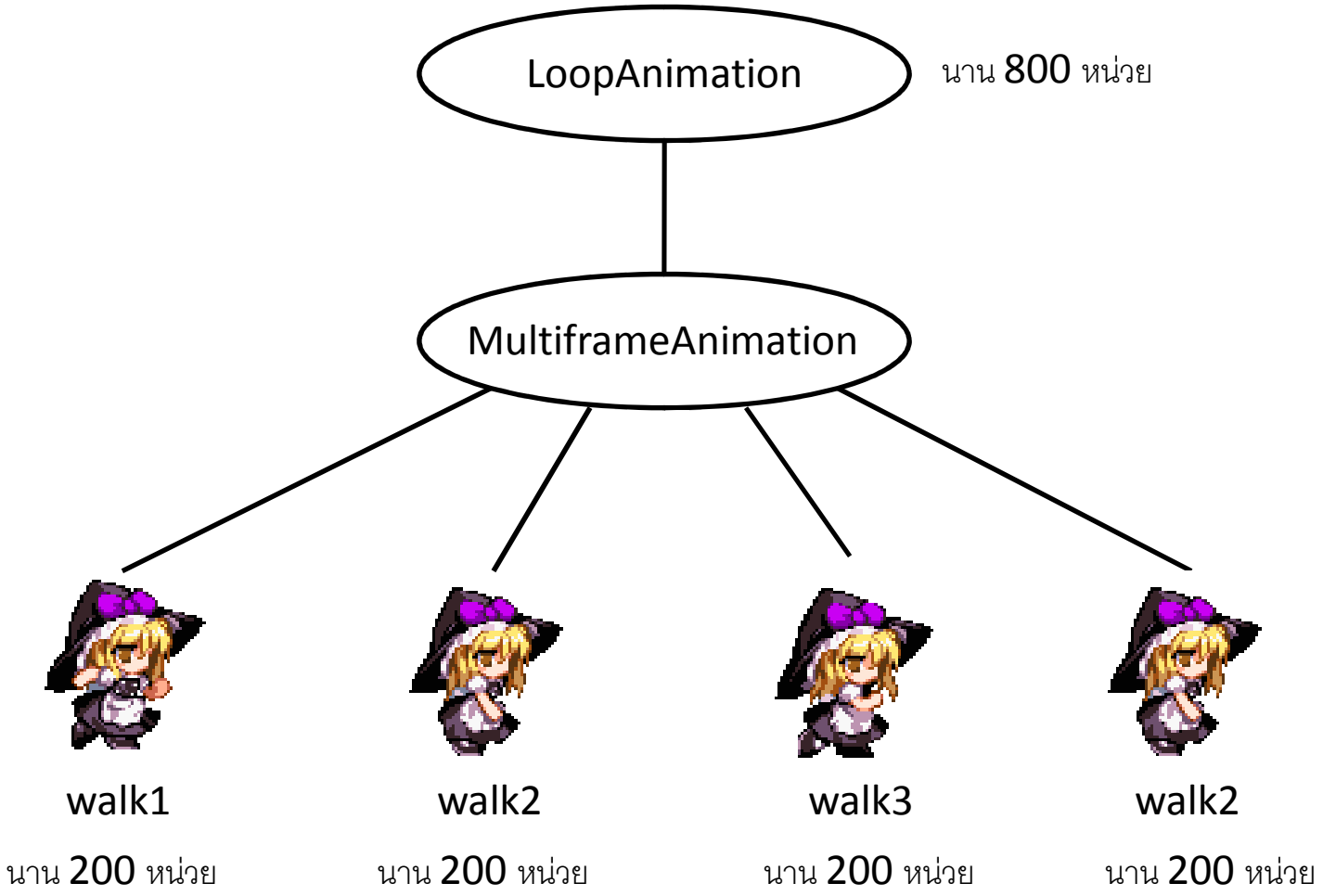
- **คำเตือน:**
  - MultiframeAnimation จะเริ่มเล่นที่เวลา 0
  - ถ้าเวลาเกินความยาวรวมของ frame ทั้งหมด (ในที่นี้คือ 800 มิลลิวินาที) รูปจะคงอยู่ที่เฟรมสุดท้าย

LoopAnimation

- ทำให้การแสดงผล animation ย้อยกลับไปเริ่มต้นใหม่เมื่อเวลาครบคาบ (period)

```
var walkAnimation =
    new LoopAnimation(
      multiframeAnimation, ← ลูก
      800); ← คาบ
```

LoopAnimation

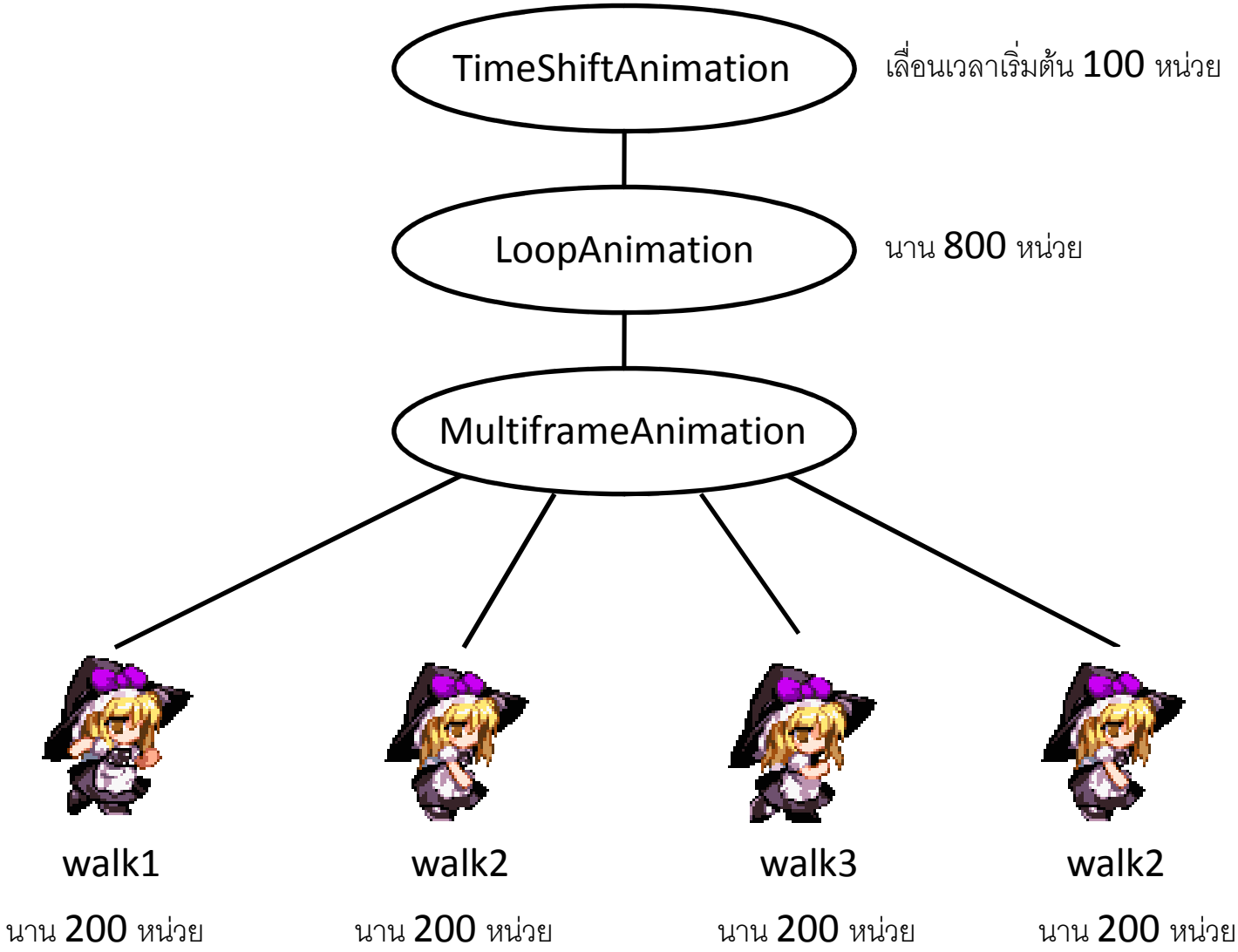


TimeShiftAnimation

- ใช้ทำให้เวลาเริ่มต้น animation เลื่อนออกไป

```
var timeShiftAnimation = new
  TimeShiftAnimation(
     walkAnimation, ← ลูก
     100); ← เวลาเลื่อน
```

TimeShiftAnimation

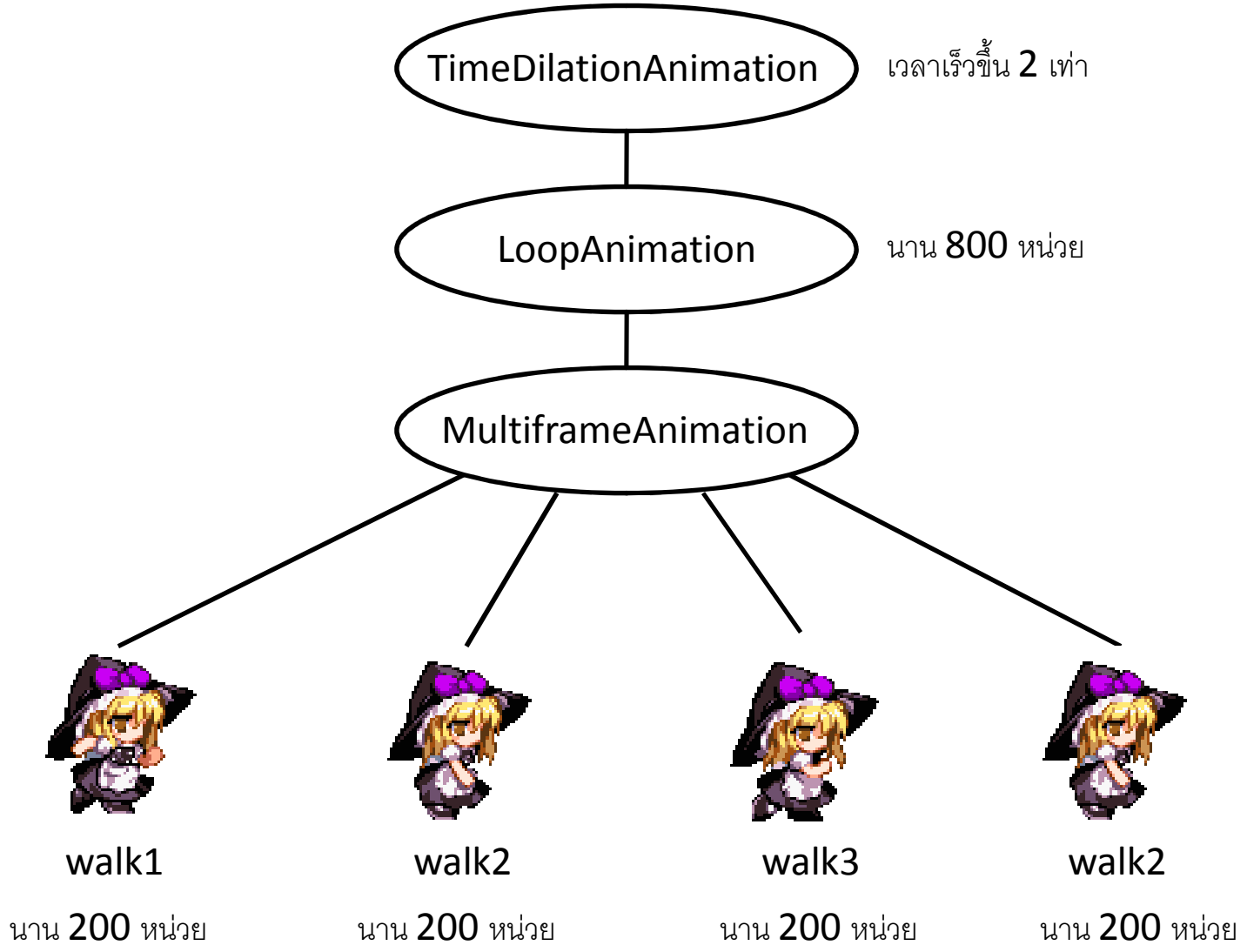


TimeDilationAnimation

- ใช้ทำให้เวลาเร็วขึ้นหรือช้าลง

```
var timeDilationAnimation = new
    TimeDilationAnimation(
      walkAnimation, ← ลูก
      2); ← เวลาเร็วขึ้นกี่เท่า
```

TimeDilationAnimation



StatefulAnimation

- ใช้สร้าง **animation** ที่มี “สถานะ”
- สถานะแต่ละสถานะมี “ชื่อ” เป็นตัวแปรประเภท **string**
- เราสามารถใช้สถานะมาสร้าง “ท่าทาง” ของตัวละครได้

stand

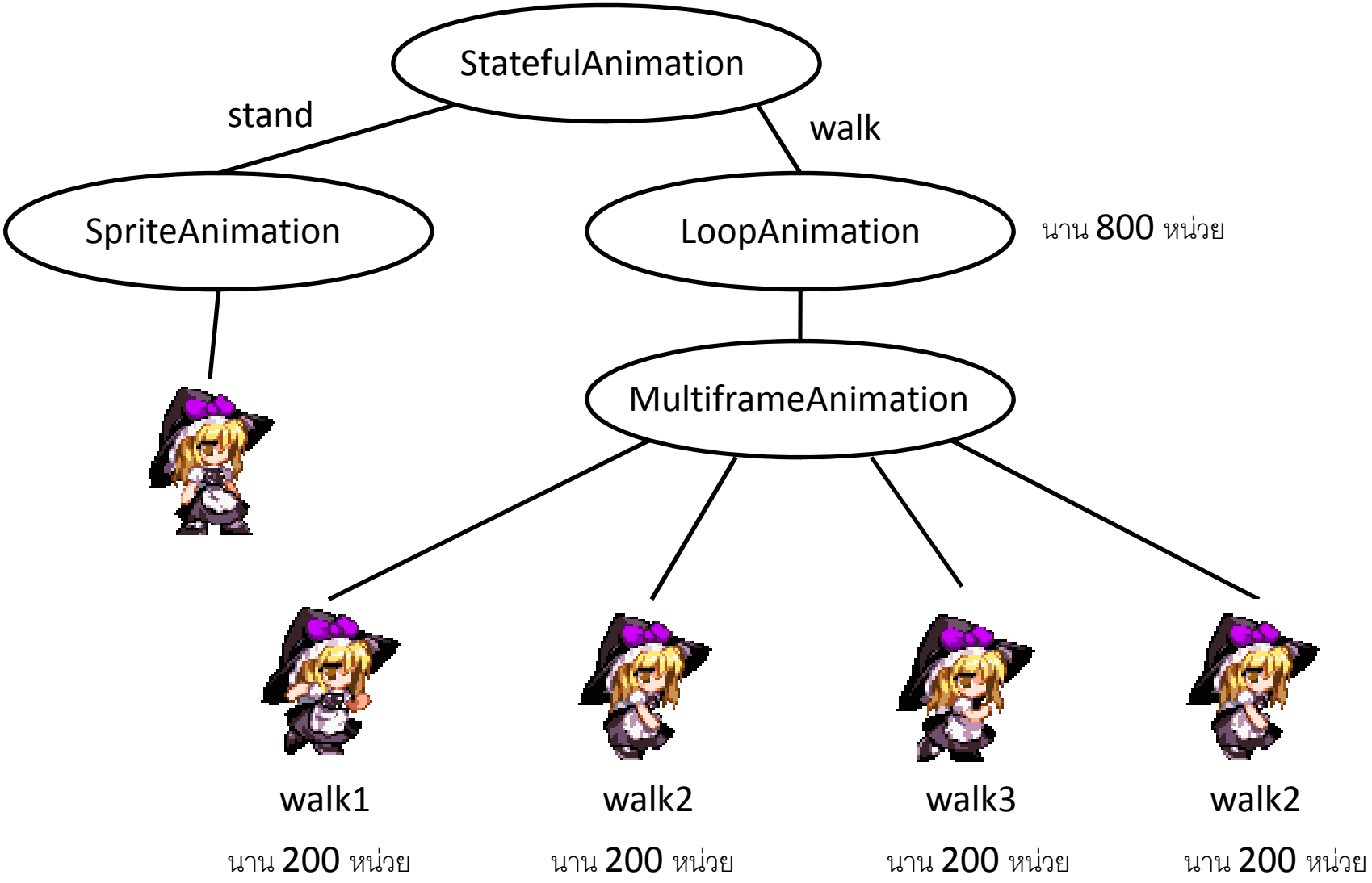
walk

StatefulAnimation

- สร้าง

```
var statefulAnimation =
    StatefulAnimation();
statefulAnimation.Set(
    "stand", ← ชื่อสถานะ
    standAnimation); ← animation ของสถานะนั้น
statefulAnimation.Set(
    "walk",
    walkAnimation);
```

StatefulAnimation



StatefulAnimation

- กำหนดสถานะปัจจุบัน

```
statefulAnimation.CurrentStateName = "walk";
```

หรือ

```
statefulAnimation.CurrentStateName = "stand";
```