“Winners Don’t Use Drugs”

William S. Sessions, Director, FBI

INSERT COIN TO PLAY

The Get Bonus

infinite entertainment system

The

Get Bonus

infinite entertainment system

Level 1

GBPPU a.k.a. Graphics

Level 2

Multiverse a.k.a Architecture

Level 3

Enumerator a.k.a. Design

The Get Bonus Picture-Processing Unit

The Get Bonus Picture-Processing Unit

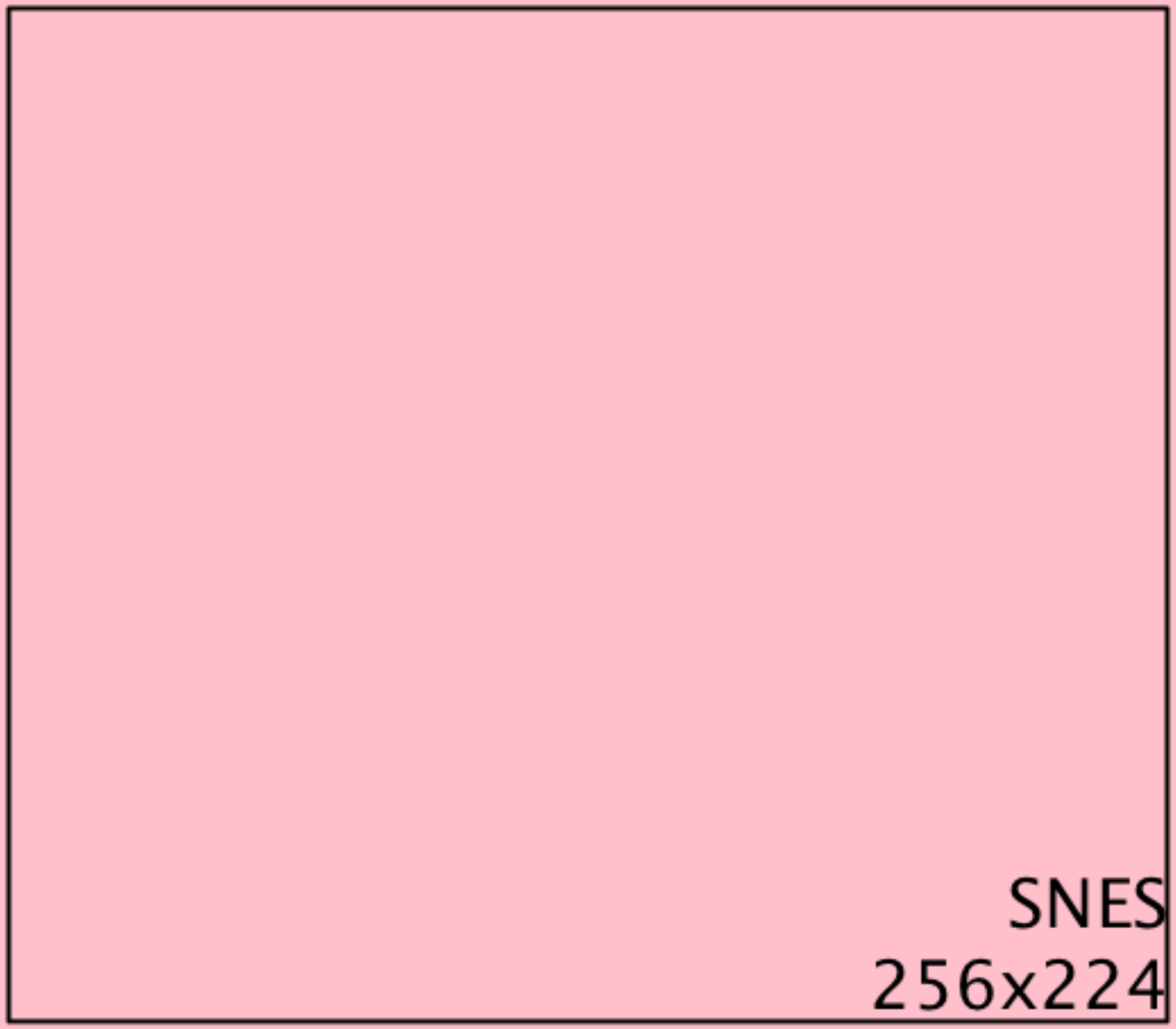
The Get Bonus Picture-Processing Unit

60 FPS

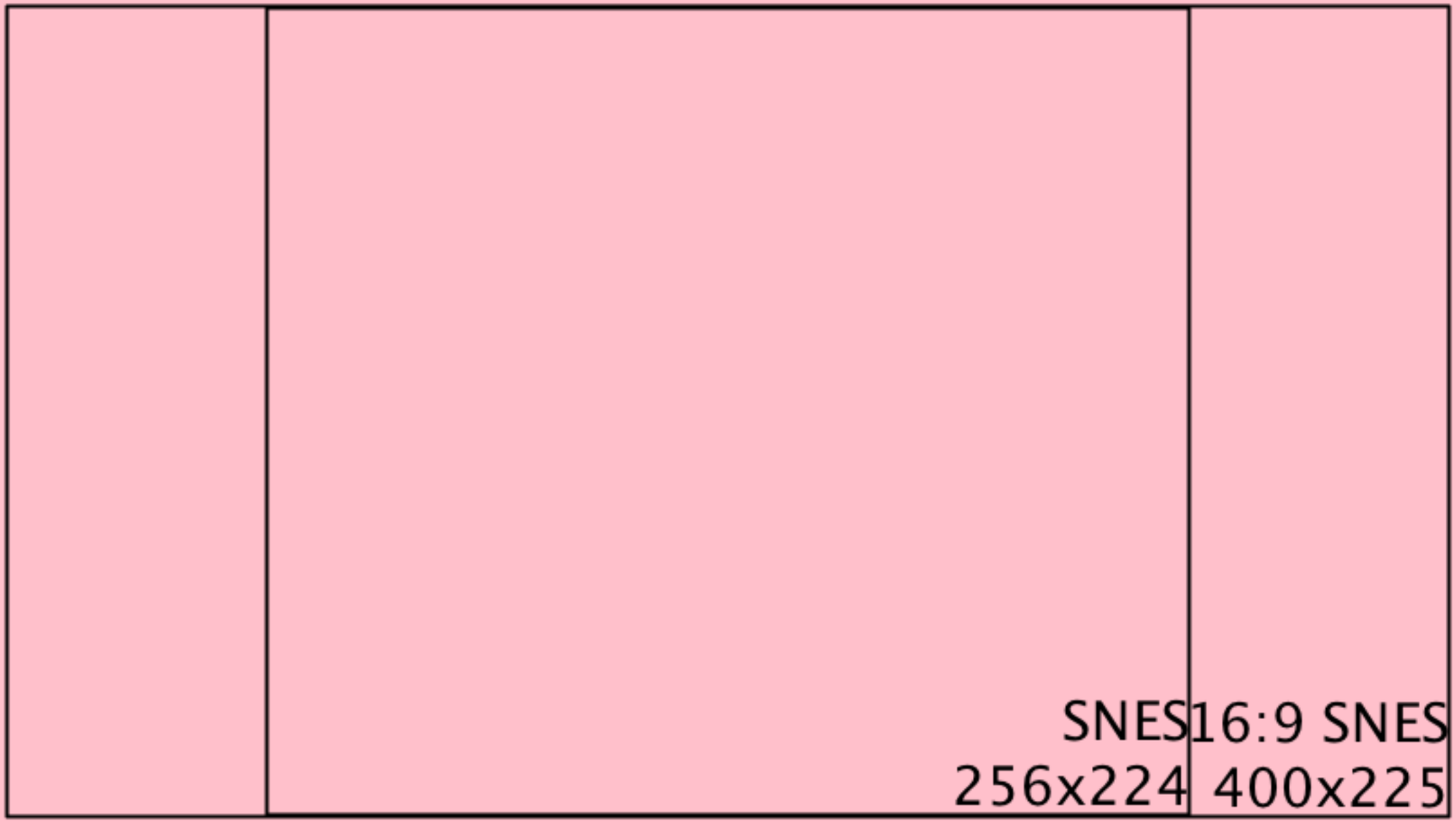
The Get Bonus Picture-Processing Unit

16.(6) ms

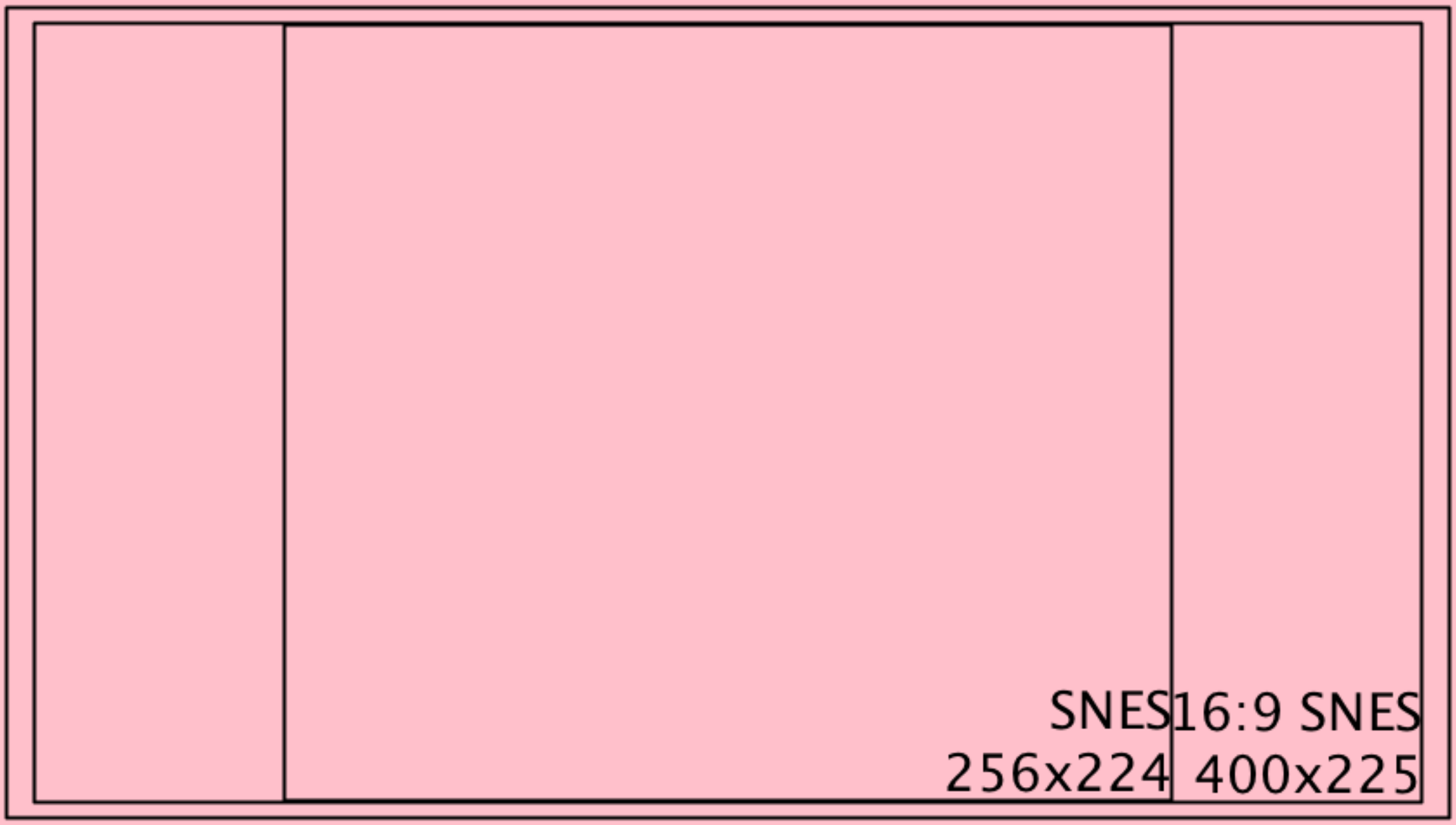
The Get Bonus Picture-Processing Unit



The Get Bonus Picture-Processing Unit

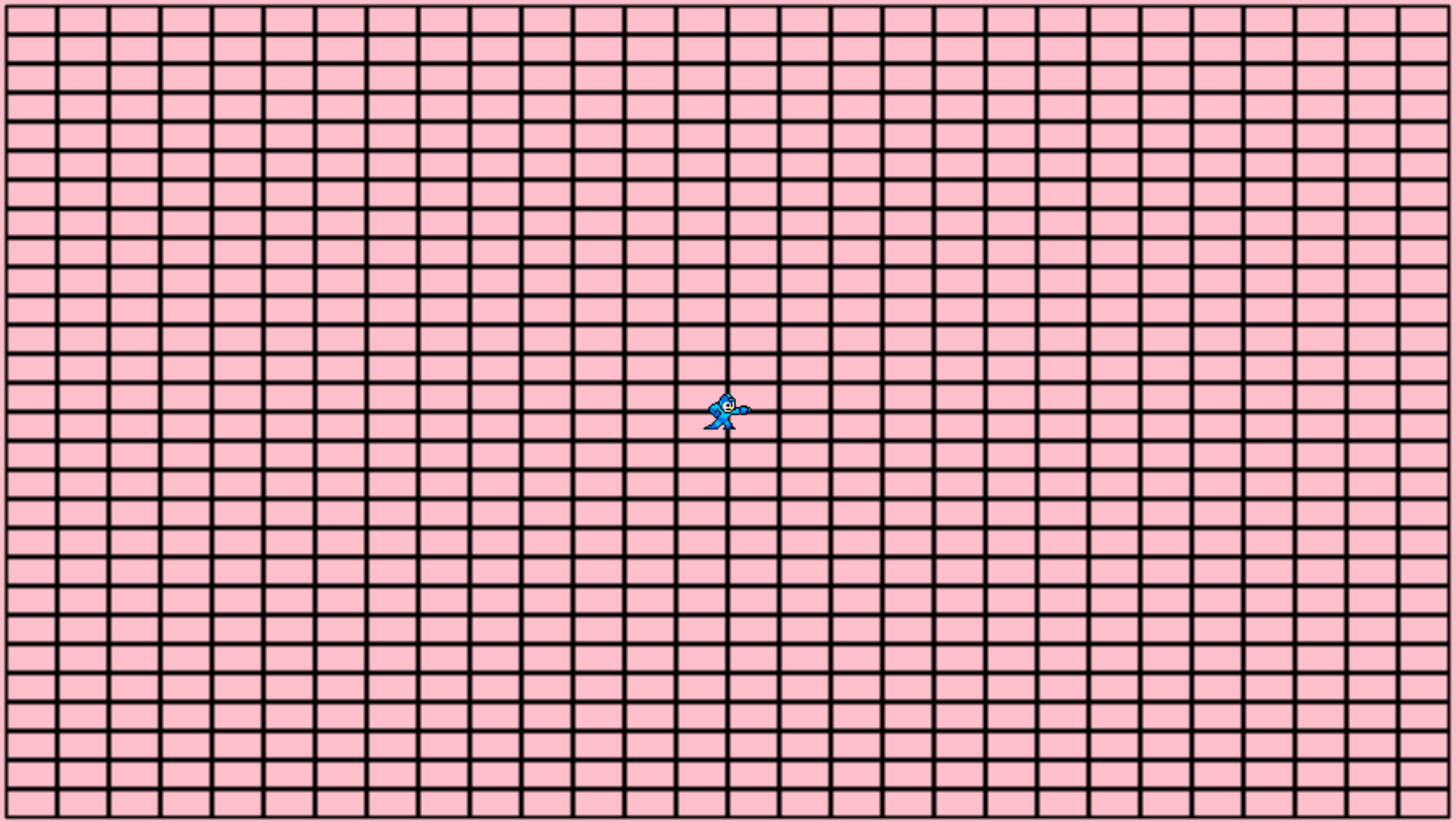


The Get Bonus Picture-Processing Unit

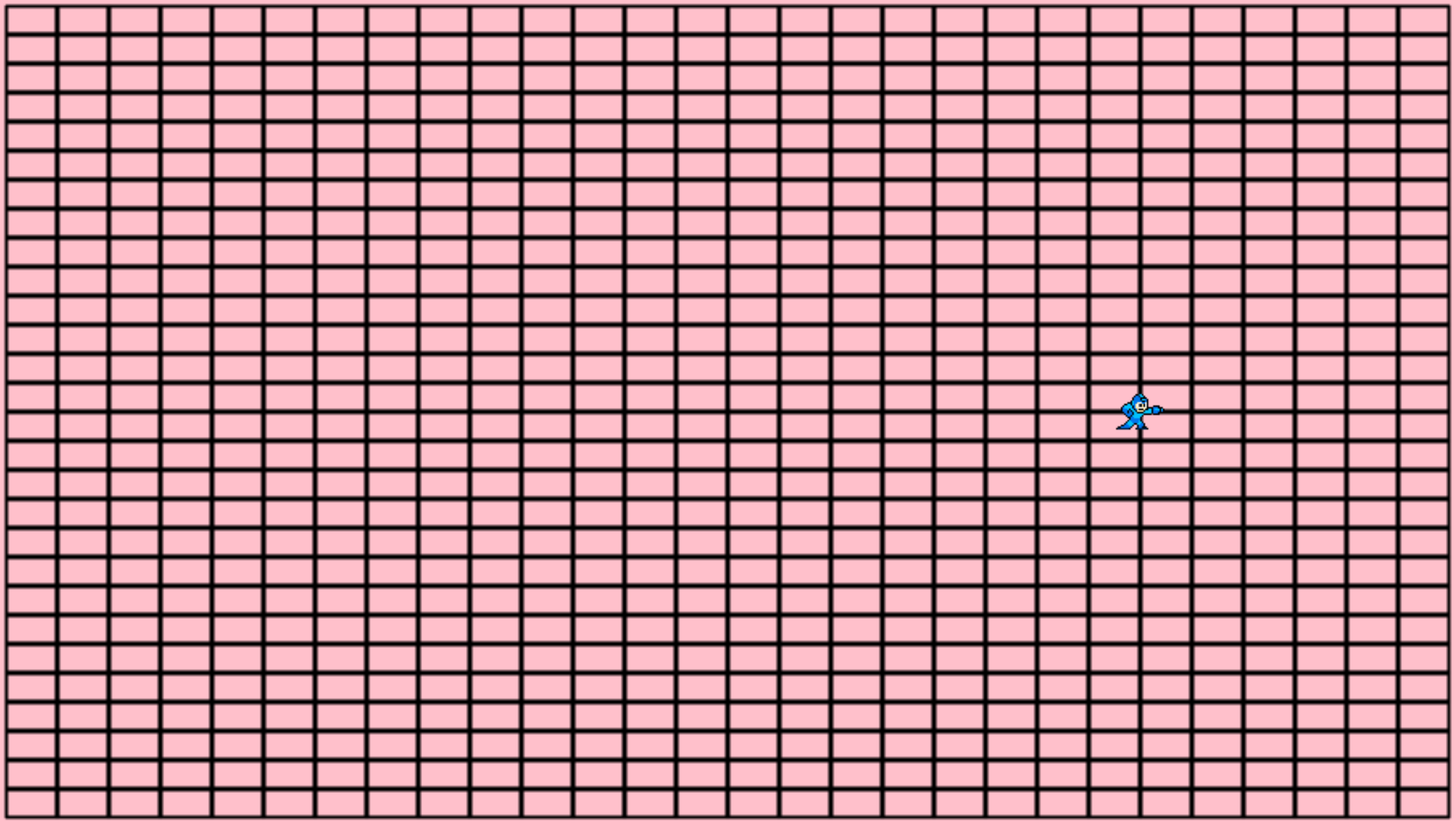


16:9 at 26 scale

The Get Bonus Picture-Processing Unit

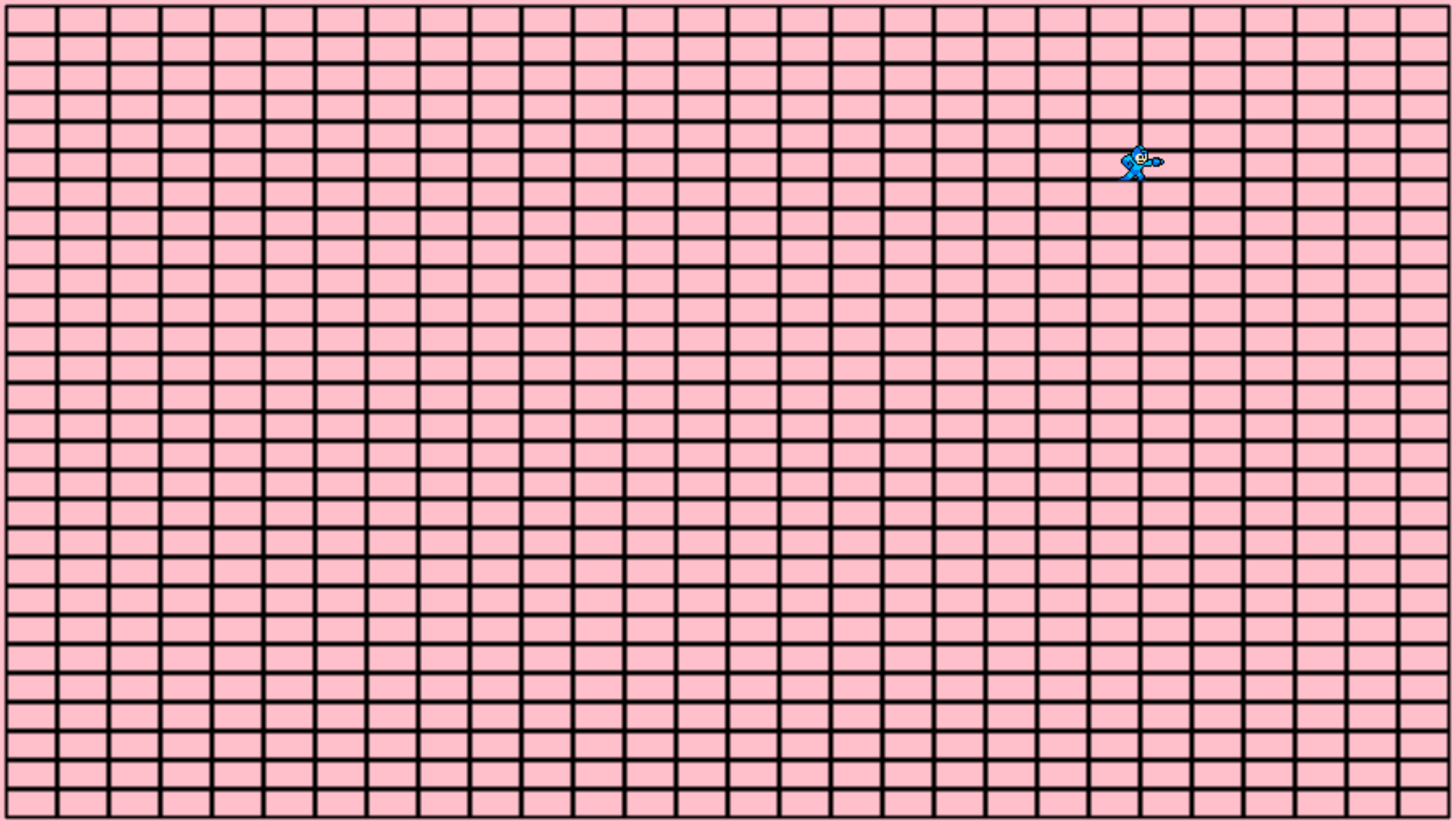


The Get Bonus Picture-Processing Unit



dx

The Get Bonus Picture-Processing Unit

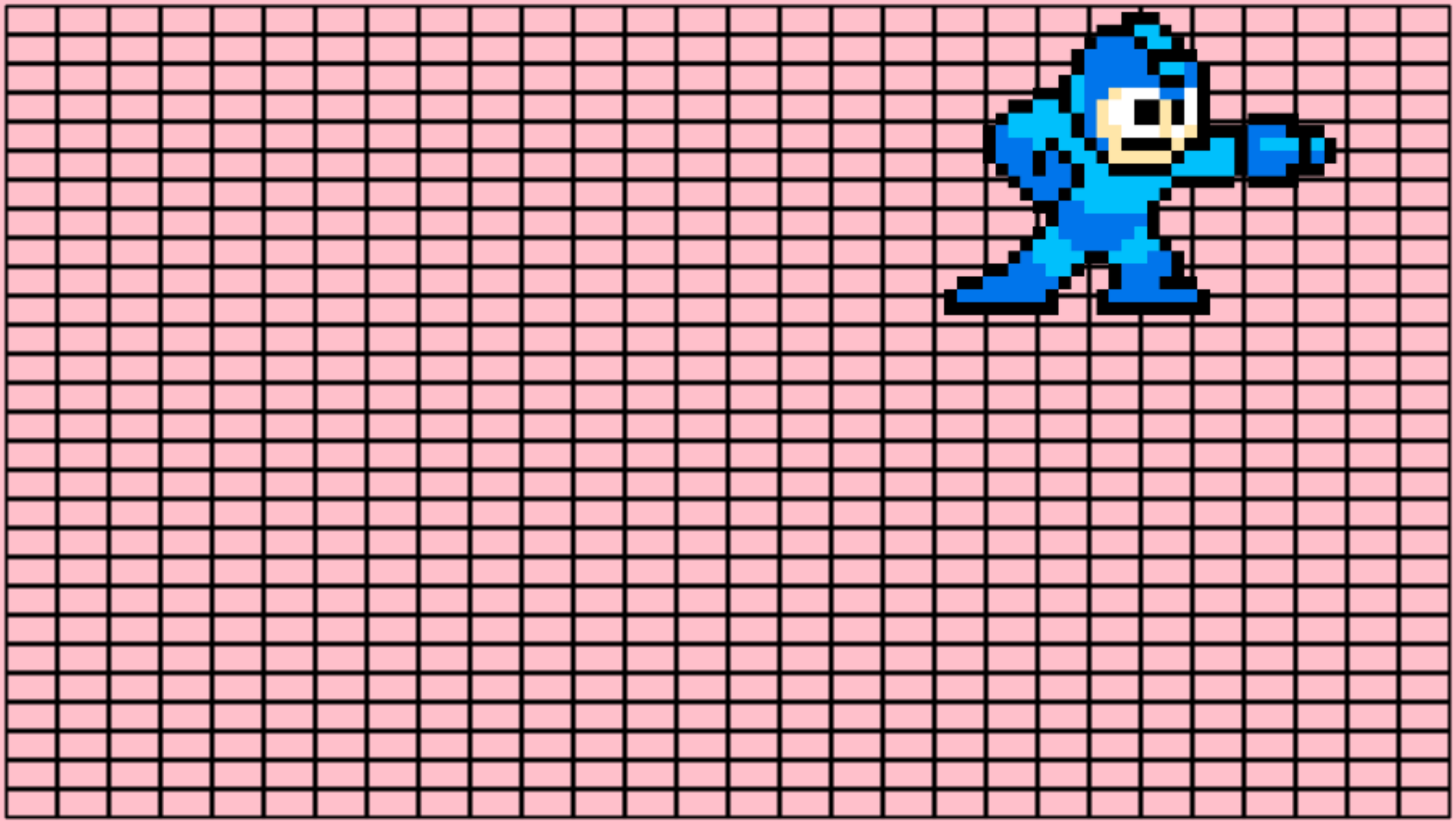


dx dy

The Get Bonus Picture-Processing Unit

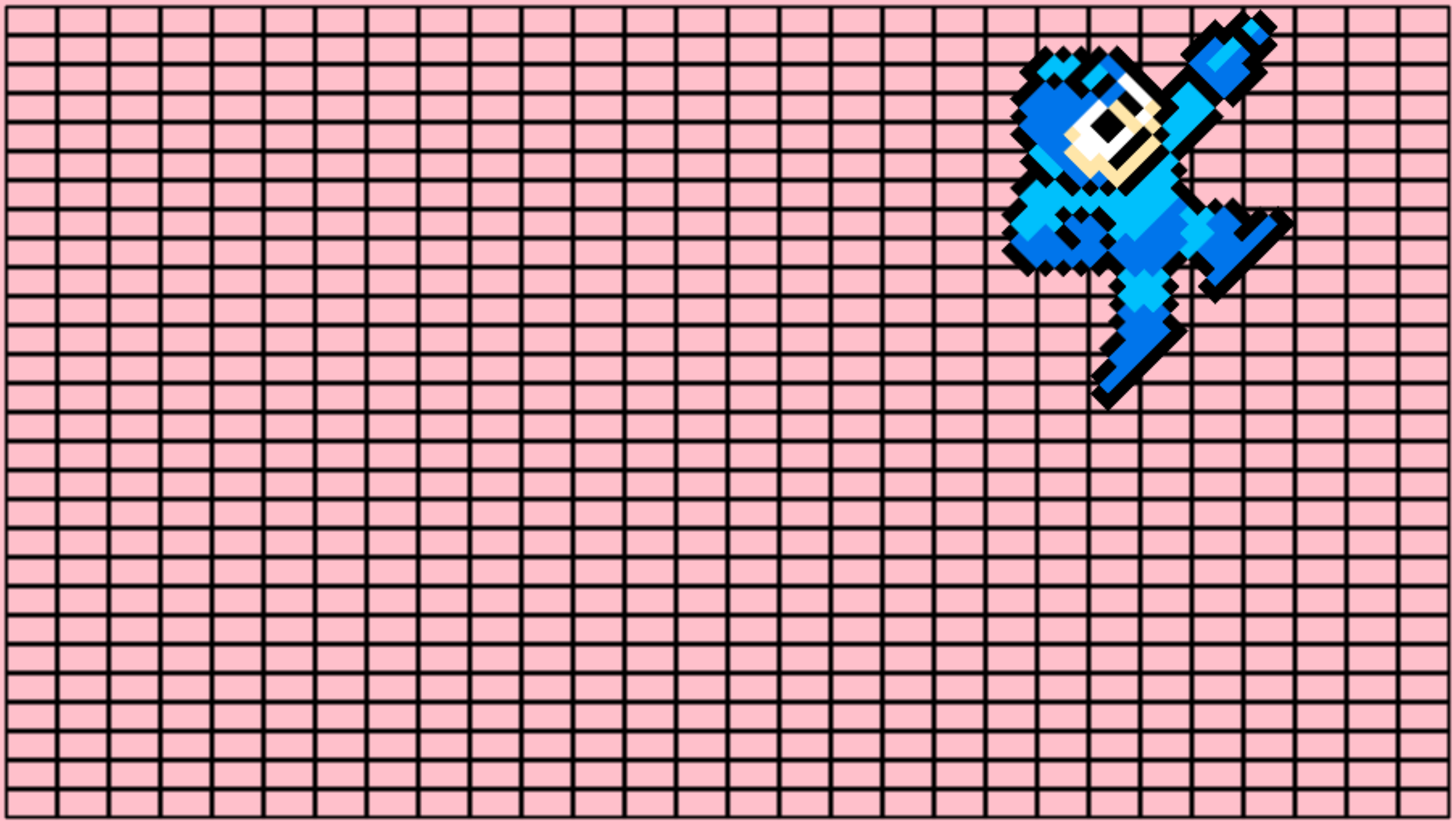
dx dy mx

The Get Bonus Picture-Processing Unit



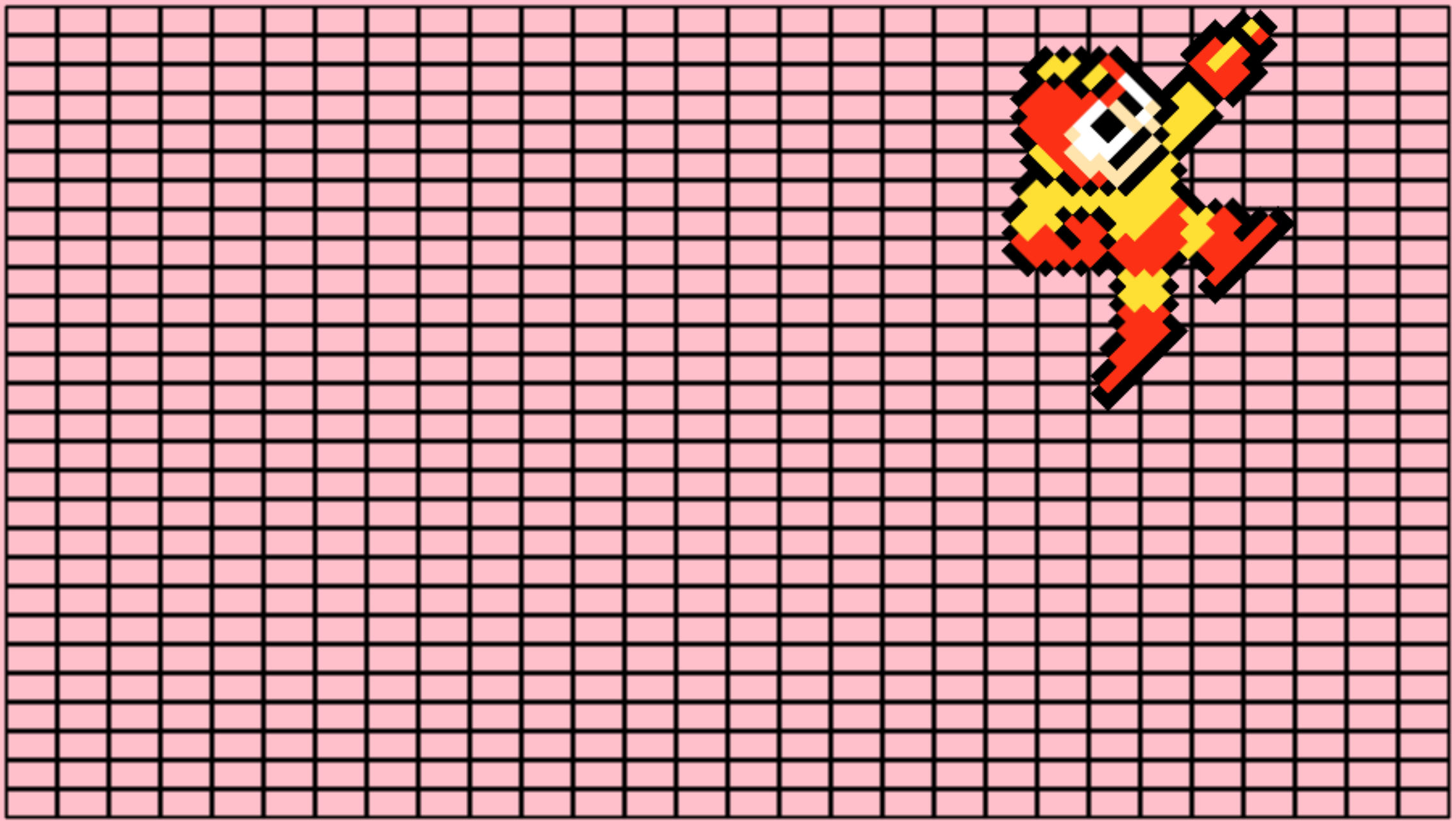
dx dy mx my

The Get Bonus Picture-Processing Unit



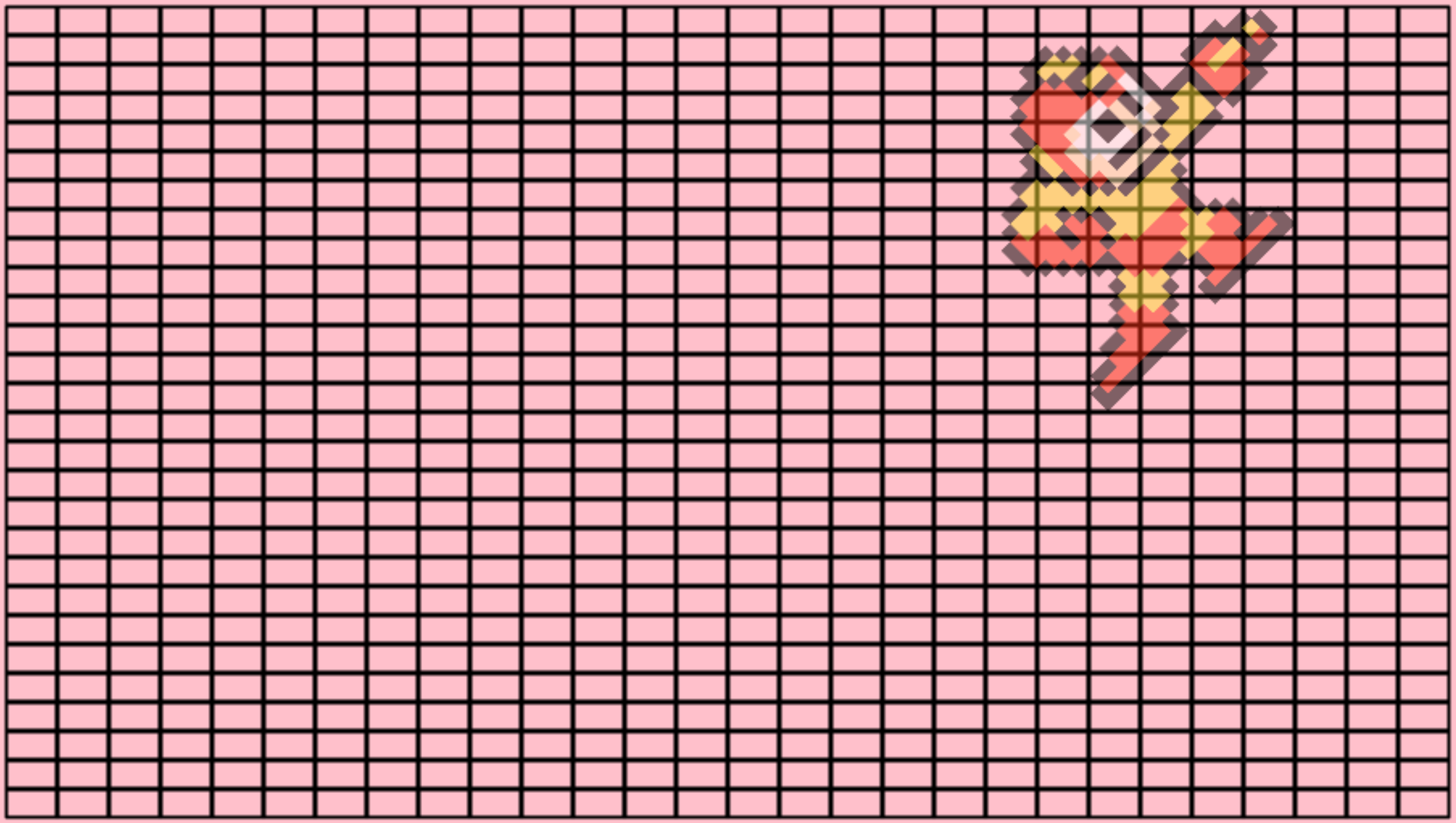
dx dy mx my θ

The Get Bonus Picture-Processing Unit



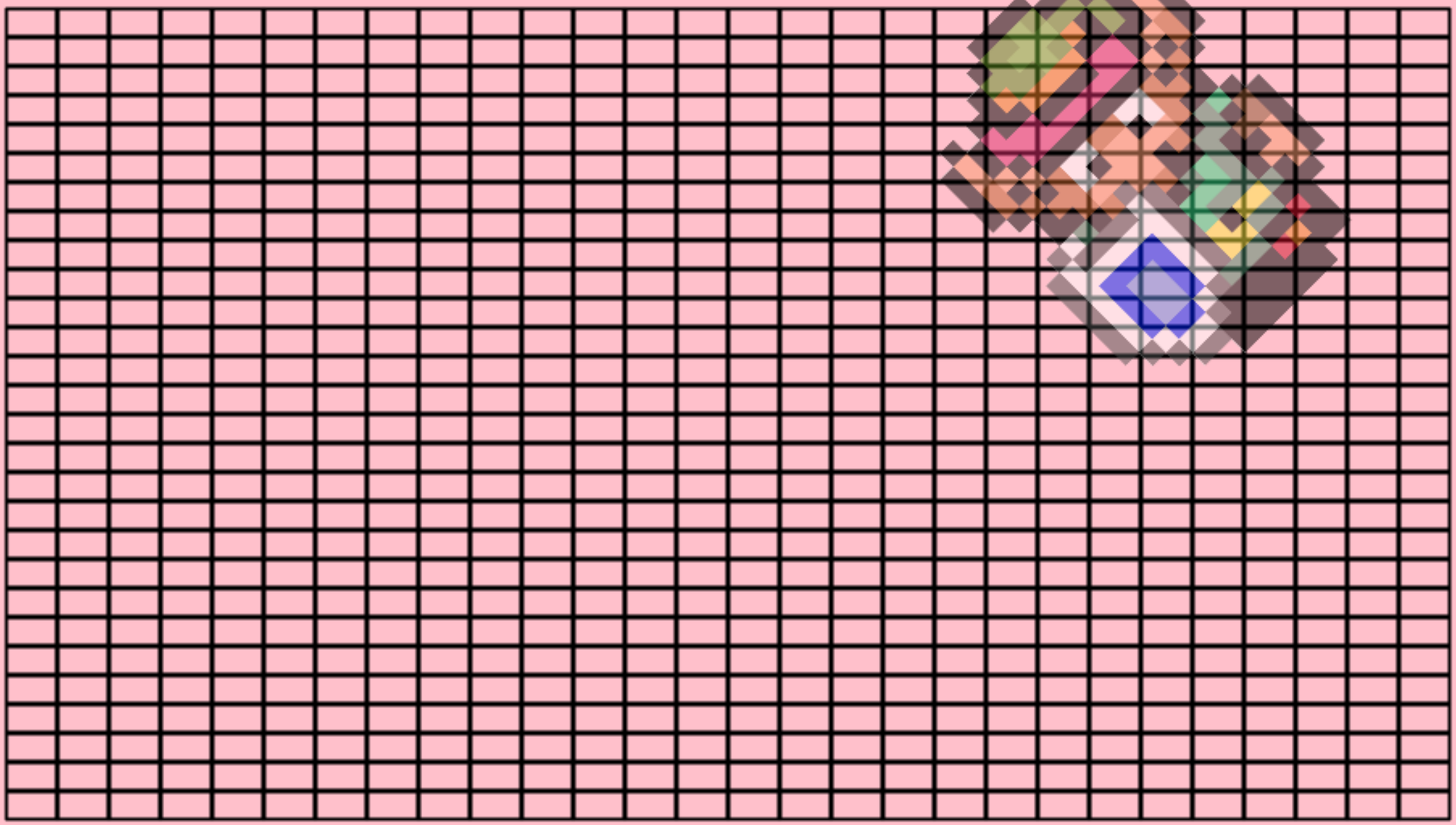
dx dy mx my θ pal

The Get Bonus Picture-Processing Unit



dx dy mx my θ pal rgba

The Get Bonus Picture-Processing Unit



dx dy mx my θ pal rgba spr

The Get Bonus Picture-Processing Unit

dx dy mx my θ pal rgba spr

The Get Bonus Picture-Processing Unit

dx dy mx my θ pal rgba spr v h x4

The Get Bonus Picture-Processing Unit

dx dy mx my θ pal rgba spr v h x3

The Get Bonus Picture-Processing Unit

dx dy mx my θ pal rgba spr v h x6

The Get Bonus Picture-Processing Unit

```
(define-cstruct sprite-info
  ([ dx  float] [ dy  float]
   [ hw  float] [ hh  float]
   [  r  uint8] [  g  uint8]
   [  b  uint8] [  a  uint8]
   [ mx  float] [ my  float]
   [  θ  float]
   [pal uint16] [spr uint16]
   [  h  sint8] [  v  sint8]))
```

The Get Bonus Picture-Processing Unit

```
(define-cstruct sprite-info
  ([ dx  float] [ dy  float]
   [ hw  float] [ hh  float]
   [  r  uint8] [  g  uint8]
   [  b  uint8] [  a  uint8]
   [ mx  float] [ my  float]
   [  θ  float]
   [pal uint16] [spr uint16]
   [  h  sint8] [  v  sint8]))
```

```
(make-cvector*
 (glMapBufferRange
  GL_ARRAY_BUFFER
  0
  (* HowManySprites
     VertsPerSprite
     (ctype-sizeof sprite-info))
  (bitwise-ior
   GL_MAP_INVALIDATE_RANGE_BIT
   GL_MAP_INVALIDATE_BUFFER_BIT
   GL_MAP_WRITE_BIT))
 sprite-info
 (* HowManySprites
    VertsPerSprite))
```

The Get Bonus Picture-Processing Unit

```
(define-cstruct sprite-info
  ([ dx  float] [ dy  float]
   [ hw  float] [ hh  float]
   [  r  uint8] [  g  uint8]
   [  b  uint8] [  a  uint8]
   [ mx  float] [ my  float]
   [  θ  float]
   [pal uint16] [spr uint16]
   [  h  sint8] [  v  sint8]))
```

```
(make-cvector*
 (glMapBufferRange
  GL_ARRAY_BUFFER
  0
  (* HowManySprites
     VertsPerSprite
     (ctype-sizeof sprite-info))
  (bitwise-ior
   GL_MAP_INVALIDATE_RANGE_BIT
   GL_MAP_INVALIDATE_BUFFER_BIT
   GL_MAP_WRITE_BIT))
 sprite-info
 (* HowManySprites
    VertsPerSprite))
```

```
(glDrawArrays
 GL_TRIANGLES 0
 (* HowManySprites
    VertsPerSprite))
```

The Get Bonus Picture-Processing Unit

```
(define-cstruct sprite-info
  ([ dx  float] [ dy  float]
   [ hw  float] [ hh  float]
   [  r  uint8] [  g  uint8]
   [  b  uint8] [  a  uint8]
   [ mx  float] [ my  float]
   [  θ  float]
   [pal uint16] [spr uint16]
   [  h  sint8] [  v  sint8]))
```

```
(make-cvector*
 (glMapBufferRange
  GL_ARRAY_BUFFER
  0
  (* HowManySprites
     VertsPerSprite
     (ctype-sizeof sprite-info))
  (bitwise-ior
   GL_MAP_INVALIDATE_RANGE_BIT
   GL_MAP_INVALIDATE_BUFFER_BIT
   GL_MAP_WRITE_BIT))
 sprite-info
 (* HowManySprites
    VertsPerSprite))
```

```
(glDrawArrays
 GL_TRIANGLES 0
 (* HowManySprites
    VertsPerSprite))
```

```
vec4 sprd =
 texelFetch(Sprites, ivec2(0, spr), 0);
Color = rgba / 255.0;
gl_Position =
 vec4(h * hw * mx,
      v * hh * my,
      0.0, 1.0)
 * glRotate(θ, 0.0, 0.0, 1.0)
 * glTranslate(dx, dy, 0.0)
 * glOrtho(0.0, 16*28, 0.0, 9*28, 1.0, -1.0);
TexCoord =
 vec2(sprd.x + ((h + 1.0)/+2.0) * sprd.z,
      sprd.y + ((v - 1.0)/-2.0) * sprd.w);
Palette = pal;
```

The Get Bonus Picture-Processing Unit

```
(define-cstruct sprite-info
  ([ dx  float] [ dy  float]
   [ hw  float] [ hh  float]
   [  r  uint8] [  g  uint8]
   [  b  uint8] [  a  uint8]
   [ mx  float] [ my  float]
   [  θ  float]
   [pal uint16] [spr uint16]
   [  h  sint8] [  v  sint8]))
```

```
(make-cvector*
 (glMapBufferRange
  GL_ARRAY_BUFFER
  0
  (* HowManySprites
     VertsPerSprite
     (ctype-sizeof sprite-info))
  (bitwise-ior
   GL_MAP_INVALIDATE_RANGE_BIT
   GL_MAP_INVALIDATE_BUFFER_BIT
   GL_MAP_WRITE_BIT))
 sprite-info
 (* HowManySprites
    VertsPerSprite))
```

```
(glDrawArrays
 GL_TRIANGLES 0
 (* HowManySprites
    VertsPerSprite))
```

```
vec4 sprd =
 texelFetch(Sprites, ivec2(0, spr), 0);
Color = rgba / 255.0;
gl_Position =
 vec4(h * hw * mx,
      v * hh * my,
      0.0, 1.0)
 * glRotate(θ, 0.0, 0.0, 1.0)
 * glTranslate(dx, dy, 0.0)
 * glOrtho(0.0, 16*28, 0.0, 9*28, 1.0, -1.0);
TexCoord =
 vec2(sprd.x + ((h + 1.0)/+2.0) * sprd.z,
      sprd.y + ((v - 1.0)/-2.0) * sprd.w);
Palette = pal;
```

```
ivec2 TexCoord_uv =
 ivec2(clampit(TexCoord.x), clampit(TexCoord.y));
vec4 SpriteColor =
 texelFetch(SpriteAtlasTex, TexCoord_uv, 0);
float PaletteOffset = SpriteColor.r * 256;
ivec2 PalCoord_uv = ivec2( PaletteOffset, Palette );
vec4 PaletteColor =
 texelFetch(PaletteAtlasTex, PalCoord_uv, 0 );
out_Color = Color + PaletteColor;
if ( out_Color.a == 0.0 )
  discard;
```

The Get Bonus Picture-Processing Unit

```
; Sprites       : (~1 kb)
;    Index
;  ->
;    Atlas Location
;  x Width
;  x Height
;
; SpriteAtlas   : (~10kb)
;    X
;  x Y
;  ->
;    Color ( 4-bit)
;
; PaletteAtlas : (~200b)
;    Index
;  x Color ( 4-bit)
;  ->
;    Color (32-bit)
```

The Get Bonus Picture-Processing Unit

One vertex is 40b
One sprite is 6 vertices
One sprite is 240b
One sprite @ 60 FPS is ~14kb/s

Intel HD Graphics 4000 supports
1,908,874 sprites at 60 FPS
OR
16 sprites per pixel
vs 32 per line on SNES

Multiverse

Multiverse

```
(struct world (....))
(big-bang (world ....)
          (on-tick world->world)
          (on-draw draw-world)
          (on-key world*key->world))
```

Multiverse

```
(struct world (player enemy1 enemy2))
(define (world->world w)
  (match-define (world p e1 e2) w)
  (struct-copy
   world w
   [player (player->player p)]
   [enemy1 (enemy->enemy e1)]
   [enemy2 (enemy->enemy e2)]))
```

Multiverse

```
(struct world:paused (....))
(struct world:playing (....))
(define (world*key->world w k)
  (match w
    [(world:paused ....)
     (if (start-button? k)
         ....
         ....)]
    [(world:playing ....)
     (if (start-button? k)
         ....
         ....)]))
```

Multiverse

```
(provide/contract
 [big-bang/os
  (-> (-> any) any)]
 [os/thread
  (-> (-> any) any)]
 [os/write
  ; blocks until next frame
  (->* (hash/c symbol? any/c) any)]
 [os/read
  (-> symbol? set?)]
 [os/exit
  (-> any)])
```

Multiverse

```
(os/thread player)
(os/thread enemy)
(os/thread enemy)
(let loop ()
  ...
  (os/read 'controller)
  (os/read 'bullet)
  ...
  (os/write    'gfx MEGA-MAN
               'sound BEEP-BOOP)
  (loop))
```

Multiverse

```
(let loop ([lhs-y (/ height 2.0)])
  (define lhs-dy
    (controller-ldpad-y (os/read* 'controller)))
  (define lhs-y-n
    (clamp min-paddle-y
           (+ lhs-y (* lhs-dy speed))
           max-paddle-y))
  (os/write
   (list
    (cons 'lhs-y lhs-y-n)
    (cons 'graphics
          (transform #:d
                     (+ lhs-x paddle-hw)
                     (- lhs-y-n paddle-hh)
                     (lhs-paddle)))))
  (loop lhs-y-n))
```

Multiverse

os/write blocking
os/read returns set

Multiverse

Interface

conforms to

Map-Reduce
and
Entity-Component-System

Multiverse (Sequential) Implementation

```
(define 0x80 (make-continuation-prompt-tag 'kernel))
(define (run-process-until-syscall p)
  (call-with-continuation-barrier
   (λ ()
     (call-with-continuation-prompt
      (λ () (os/exit (p)))
      0x80
      (λ (x) x)))))
(define (trap-syscall k->syscall)
  (call-with-current-continuation
   (λ (k)
     (abort-current-continuation 0x80 (k->syscall k)))
   0x80))
```

Multiverse (Sequential) Implementation

```
(struct process (pid k) #:transparent)
(struct os (cur-heap next-heap cur-procs next-procs))
(define boot
  (match-lambda
   [(os cur-h next-h
        (list) next-ps)
    (os next-h (make-hasheq) next-ps (list))]
   [(os cur-h next-h
        (list* (process pid now) cur-ps) next-ps)
    (define syscall (run-process-until-syscall now))
    (boot (syscall pid
                   (os cur-h next-h cur-ps next-ps)))]))
```

Multiverse (Sequential) Implementation

```
(define-syscalls (pid current)
  [(os/exit k v)
   current]
  [(os/read k id)
   (match-define (os cur-h next-h cur-ps next-ps) current)
   (os cur-h next-h
       (snoc* cur-ps
              (process pid (λ () (k (hash-ref cur-h id empty)))))
       next-ps)]
  [(os/write k id*vals)
   (match-define (os cur-h next-h cur-ps next-ps) current)
   (for ([id*val (in-list id*vals)])
     (match-define (cons id val) id*val)
     (hash-update! next-h id (curry cons val) (λ () empty)))
   (os cur-h next-h cur-ps
       (snoc* next-ps
              (process pid (λ () (k (void))))))]
  [(os/thread k t)
   (match-define (os cur-h next-h cur-ps next-ps) current)
   (define t-pid (gensym 'pid))
   (os cur-h next-h
       (snoc* cur-ps
              (process pid (λ () (k t-pid)))
              (process t-pid t))
       next-ps)])
```

Enumerator

Enumerator

Play : Skill -> Score

Enumerator

Play : deterministic! Score

Enumerator

Play : Skill -> Score

Score = Bool

Enumerator

Play : Skill -> Score

Score = [0,1]

Enumerator

Play : Skill -> Score

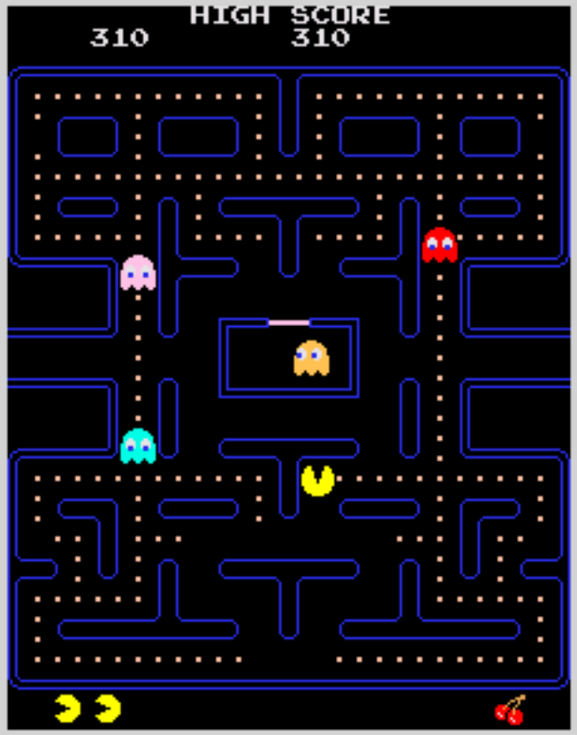
Score = Real

Enumerator

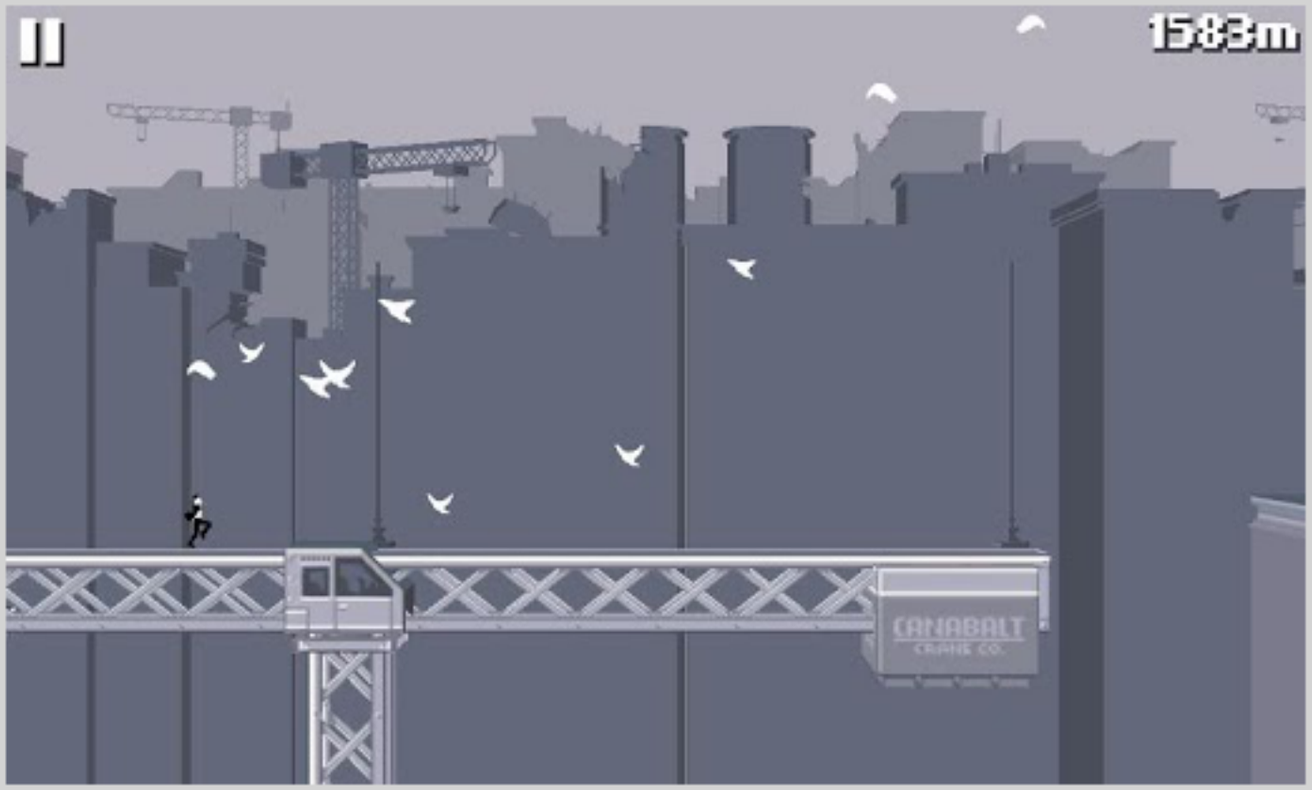
Play : Skill -> Score

Score = Score x Score

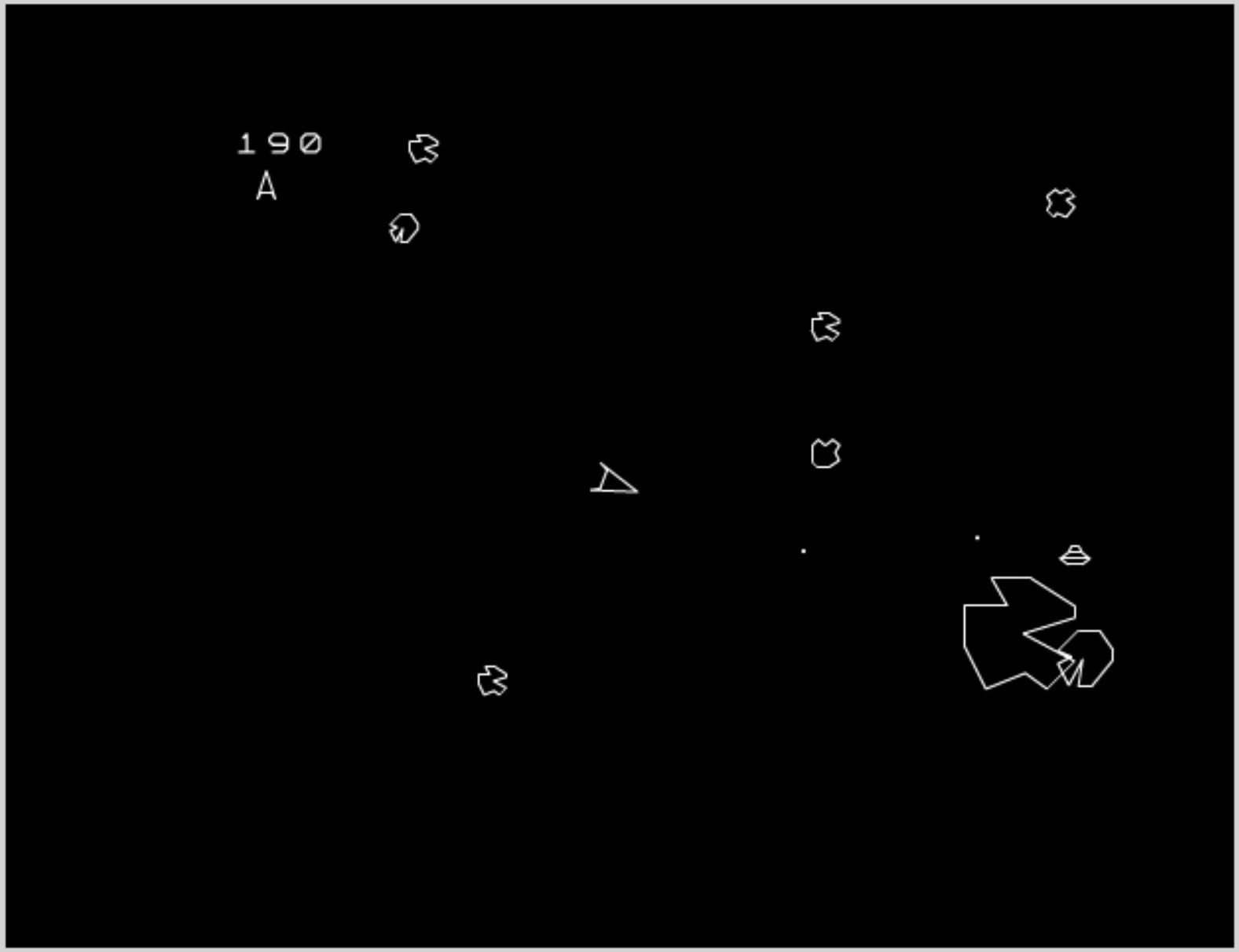
Enumerator



Enumerator



Enumerator



Enumerator

Game : Level -> Play

Enumerator

Game : Level -> Play

b = bijection(Level, Nat)

Enumerator

Game : Level -> Play

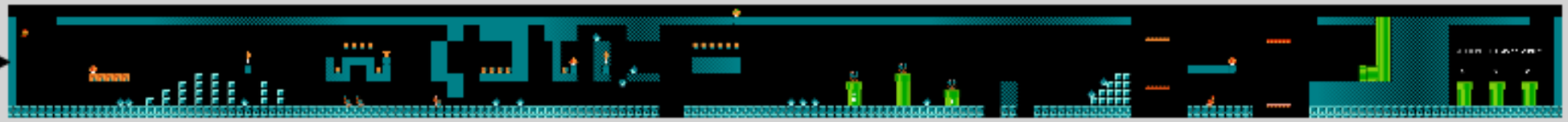
b = bij el, Nat)

not "procedural"

Enumerator

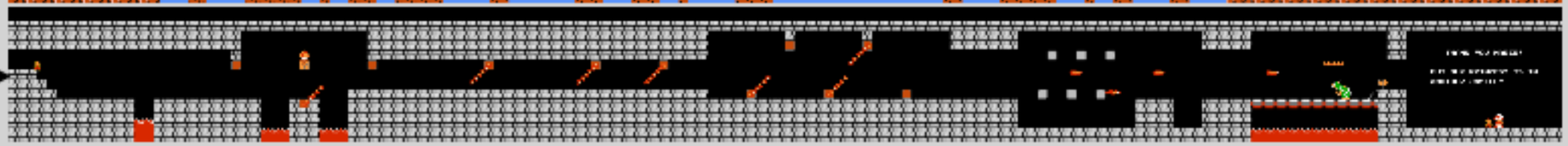
0 →

7 →



24 →

32 →



67 →

81 →

90 →

101 →

102 →

Enumerator

Game.Gen : Params -> Level

Enumerator

Game.Gen(Underwater, Easy)

Enumerator

Game.Gen(Underwater, Easy)

81

Enumerator

Game.Gen(Underwater, Easy)

81

Enumerator

Series : Version -> Game

Enumerator

Series : Version -> Game

Enumerator

Series : Version -> Game

Enumerator

Series : Version -> Game

Enumerator

Series : Version -> Game

Enumerator

Series : Version -> Game

Enumerator

Series : Version -> Game

Game.Gen : Params -> Level

Enumerator

Series : Version -> Game

Game.Gen : Params -> Level

Game : Level -> Play

Enumerator

Series : Version -> Game

Game.Gen : Params -> Level

Game : Level -> Play

Play : Skill -> Score

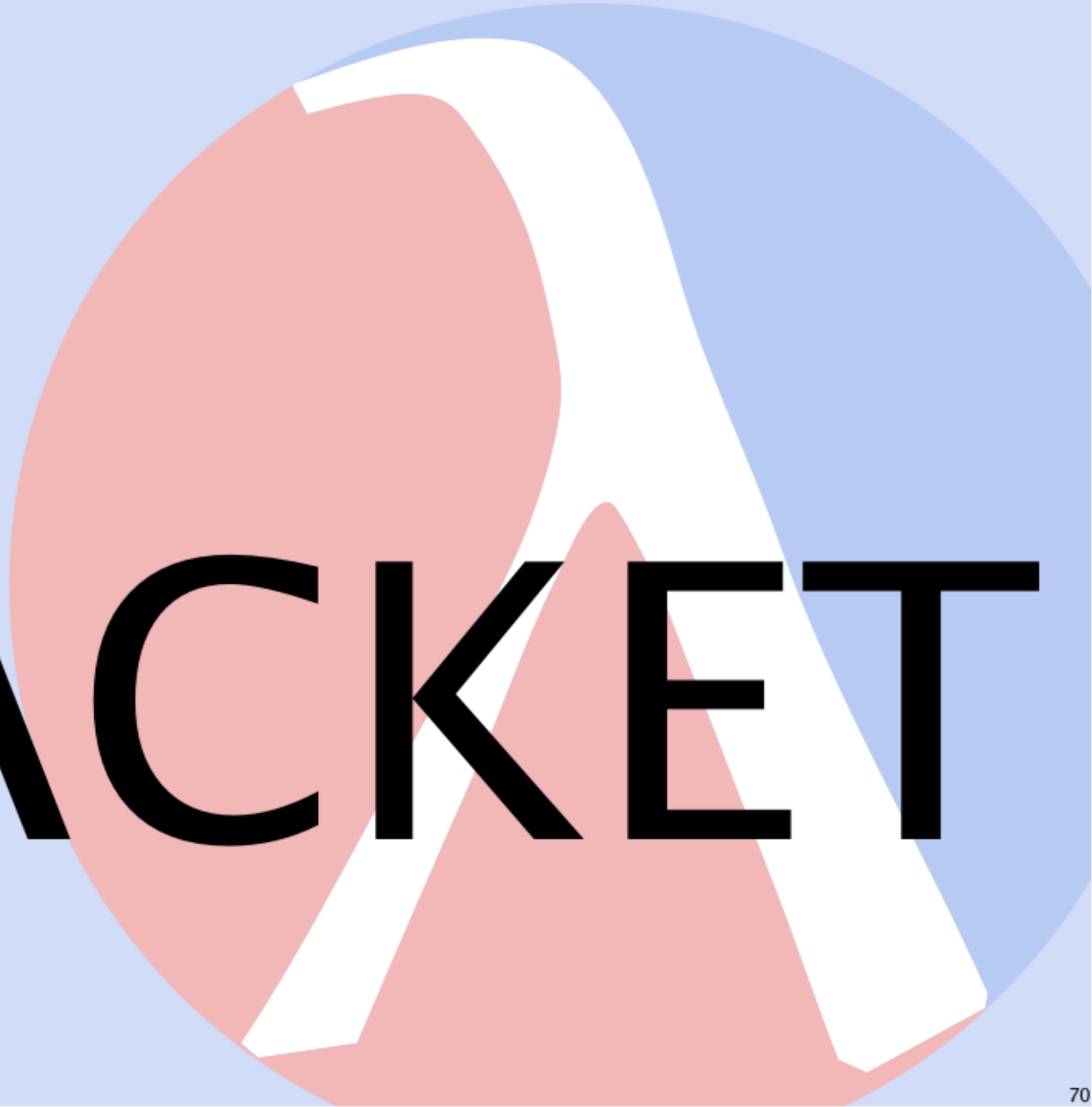
The
Get Bonus
infinite entertainment system

Level 1
GBPPU a.k.a. unsafe C interop, OpenGL, and GPU shaders

Level 2
Multiverse a.k.a delim cont-based OS and map-reduce

Level 3
Enumerator a.k.a. series, games, and Godel bijections

Now
You're
Playing
With



RACKET