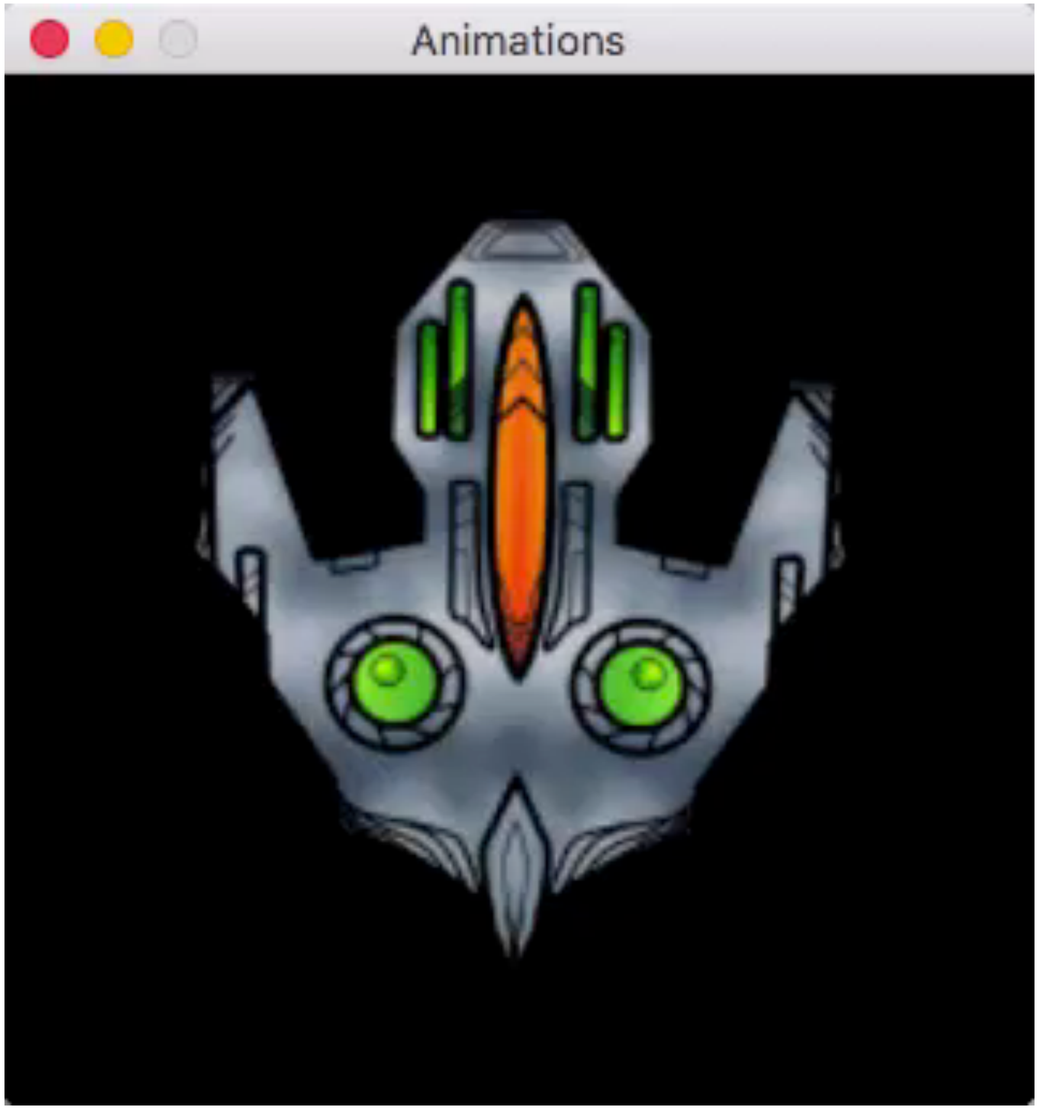
7 Programming with Animations

Reminder: Frame-by-Frame Animations

http://www.faithistorment.com/2016/03/
rotoscoped-horse-by-carnegie-mellon-university-school-of-art.html

Example: Interpolated Animations



Credits:
opengamearts.com
Technozomians

Interpolating Colors

```
red = (255, 0, 0)
blue = (0, 0, 255)
white = (255, 255, 255)

def blend_color (color1, color2, blend_factor):
    red1, green1, blue1 = color1
    red2, green2, blue2 = color2
    red0 = red1+(red2-red1)*blend_factor
    green0 = green1+(green2-green1)*blend_factor
    blue0 = blue1+(blue2-blue1)*blend_factor
    return int(red0), int(green0), int(blue0)


blend_color(red, blue, colorfactor)
```

Interpolating Colors and Size

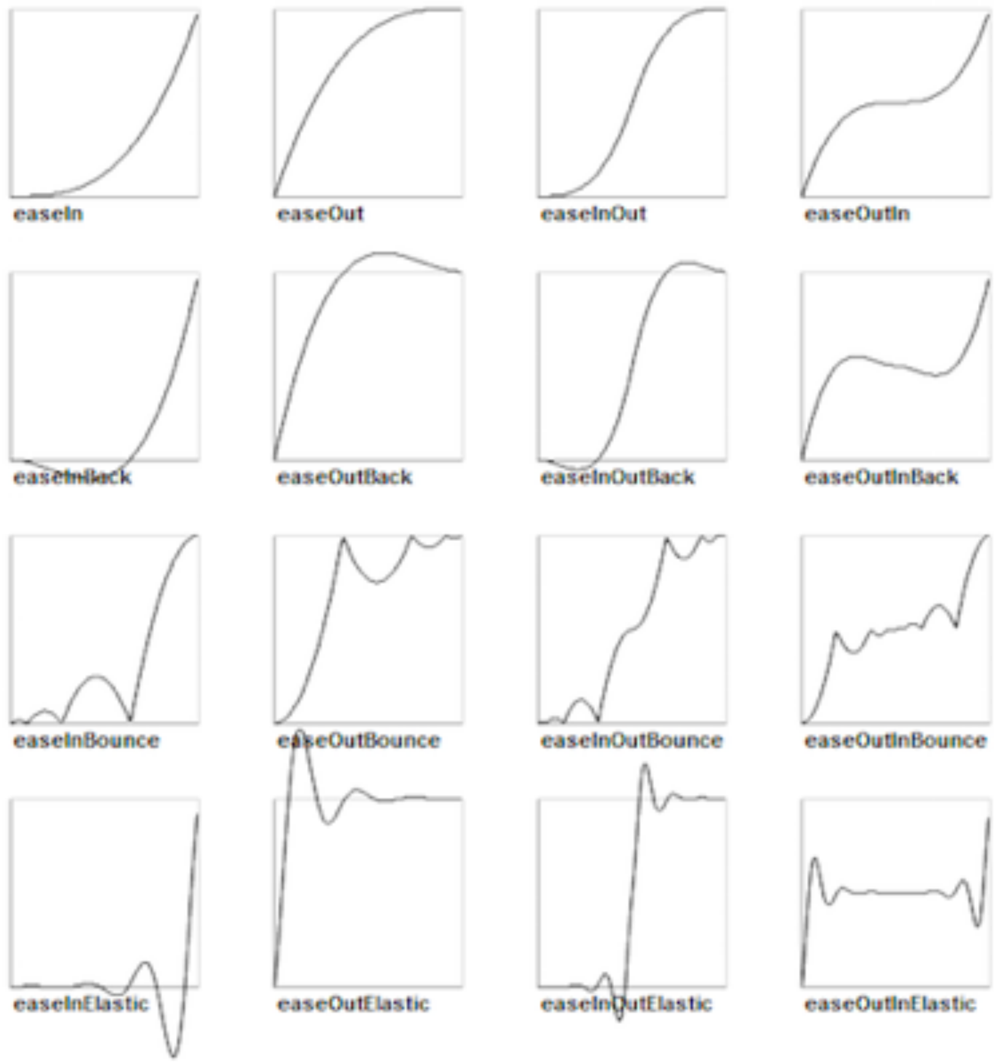
```
…
steps = (xend - xstart)/dx
dsize = 1.0/steps
dcolor = 1.0/steps
clock = pygame.time.Clock()
x = xstart
y = 240
r = 40
sizefactor = 1
colorfactor = 0

while True:
    for event in pygame.event.get():…
    xr = r*sizefactor
    if x+xr <= scr_w:
        pygame.draw.rect(scr, white, Rect((0, 0), (scr_w, scr_h)))
        pygame.draw.circle
         (scr, blend_color(red, blue, colorfactor), (x, y), round(xr))
        x += dx
        sizefactor += dsize
        colorfactor += dcolor
    timepassed_secs = clock.tick(framerate) / 1000.0
    pygame.display.update()
```

AnimationBasicsCS.py

Non-Linear Interpolation, Easing

- Plain linear interpolation does not match physical reality
  - Inertia, acceleration, deceleration, overshooting
  - Easing in/out: Gradual increase/decrease of velocity
- Custom interpolators
  - Use non-linear computation of intermediate values
- Examples from Cocos2d-x: See to the right



Example Cocos2d-x: Various Interpolators

```
auto fi = FadeIn::create(4.0f);
auto fo = FadeOut::create(4.0f);
auto sd = ScaleTo::create(4.0f, 0.1f);
auto su = ScaleTo::create(4.0f, 1.5f);
auto fisu = Spawn::createWithTwoActions(fi, su);
auto fosd = Spawn::createWithTwoActions(fo, sd);
auto mru = MoveBy::create(2.0f,Vec2(50,50));
auto mld = MoveBy::create(2.0f,Vec2(-100,-100));
auto rotr = RotateBy::create(4.0f,360.0f);
auto tintR = TintTo::create(4.0f,255.0f,0.f,0.f);
auto tintW = TintTo::create(4.0f,255.0f,255.0f,255.0f);
auto rotTintR = Spawn::createWithTwoActions(rotr, tintR);
auto rotTintW = Spawn::createWithTwoActions(rotr, tintW);
auto rotrEI = EaseBounceIn::create(rotr->clone());
auto rotrEO = EaseBounceOut::create(rotr->clone());

imageSprite->runAction(Sequence::create(
  fisu, mru, mld, mru, rotr, rotTintR, rotTintW, rotrEI, rotrEO, fosd,
  NULL));
```

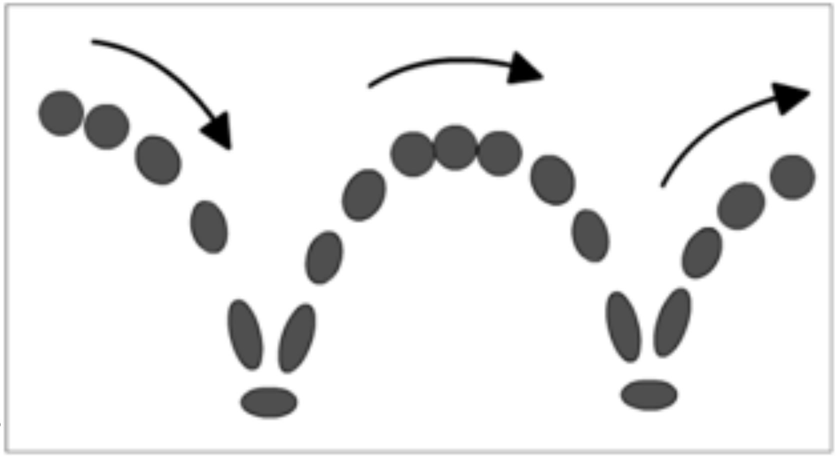
7 Programming with Animations

Literature:

K. Besley et al.: Flash MX 2004 Games Most Wanted,
Apress/Friends of ED 2004
– Section 7.4 based on book chapter 2 by ***Brad Ferguson***
T. Jones et al.: Foundation Flash Cartoon Animation,
Apress/Friends of ED 2007

Principles of (2D-)Animation

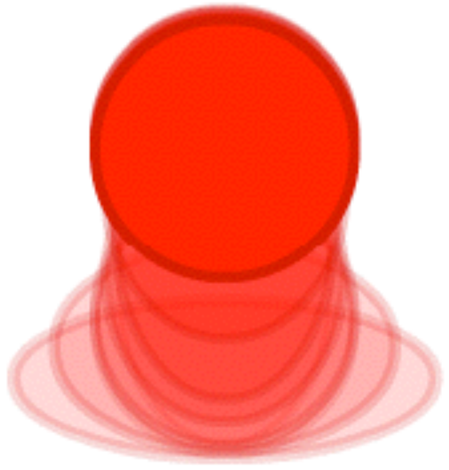
- Squash and Stretch
  - Shape of subject reacts to speed and force of movement
- Timing
  - E.g. ease-in and ease-out: Gives animation a sense of weight and gravity
- Anticipation, Action, Reaction, Overlapping Action
  - Anticipation: Build up energy before a movement
  - Reaction: Don’t simply stop, but show the process of stopping
  - Overlapping: Hierarchy of connected objects moves in a complex way
- Arcs
  - Every object follows a smooth arc of movement

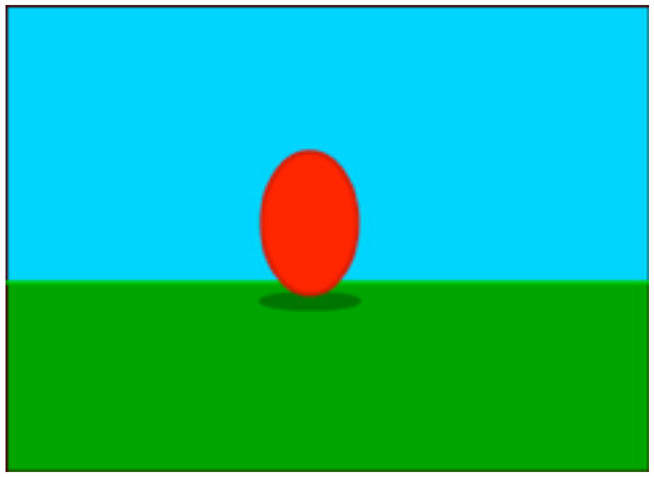


Bouncing ball
Source: T. Jones

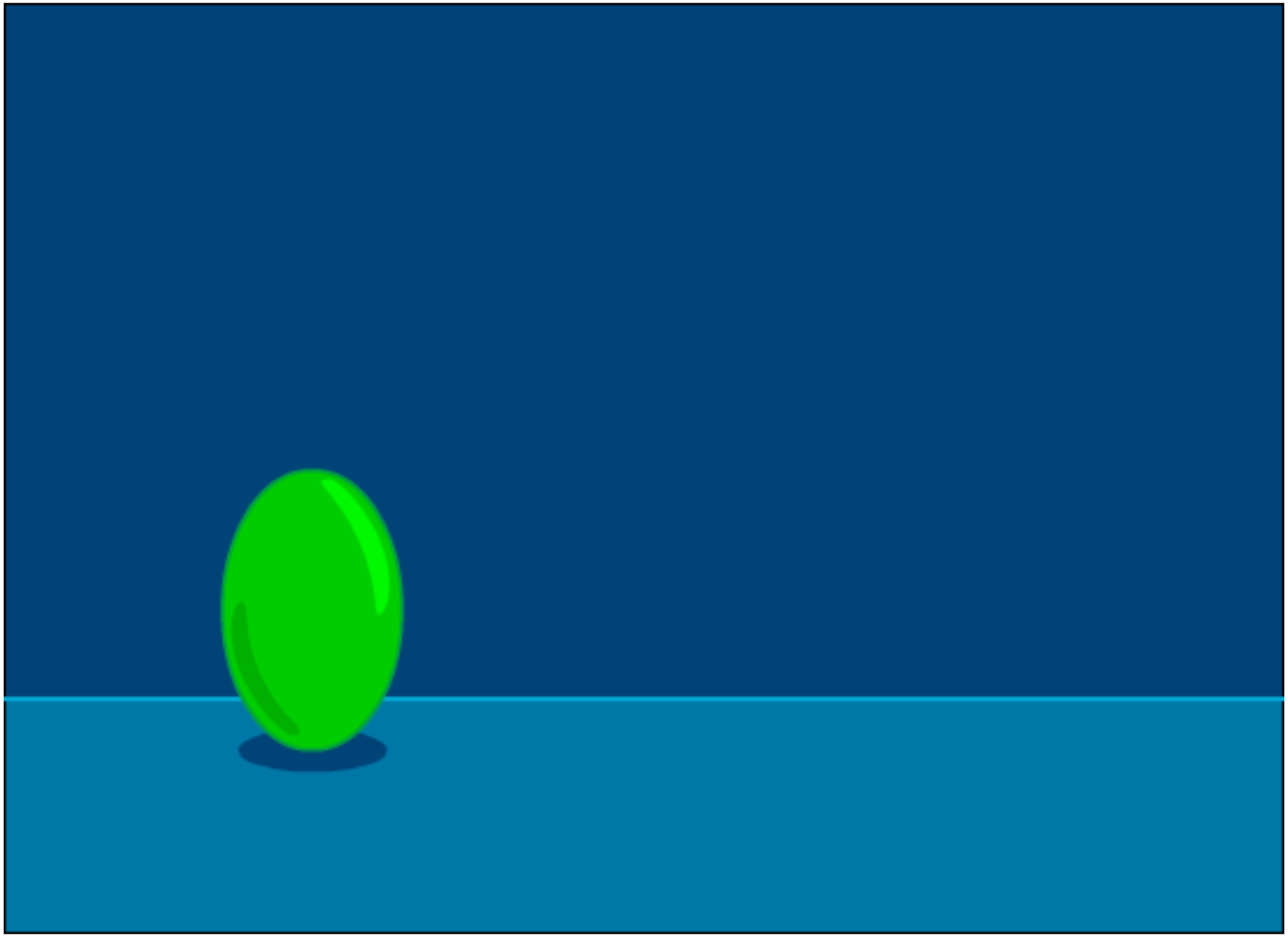
Animating a Bouncing Ball

- Going up slower than when falling
- On rise and fall, ball is stretched to give the illusion it is traveling quickly. Effect more extreme on the fall.
- At top of movement: hang time
- When ball hits the ground (and not before), it gets squashed horizontally.
- Shadow animation increases the optical illusion.
- Please note: These are exaggerations for the sake of a stronger illusion.

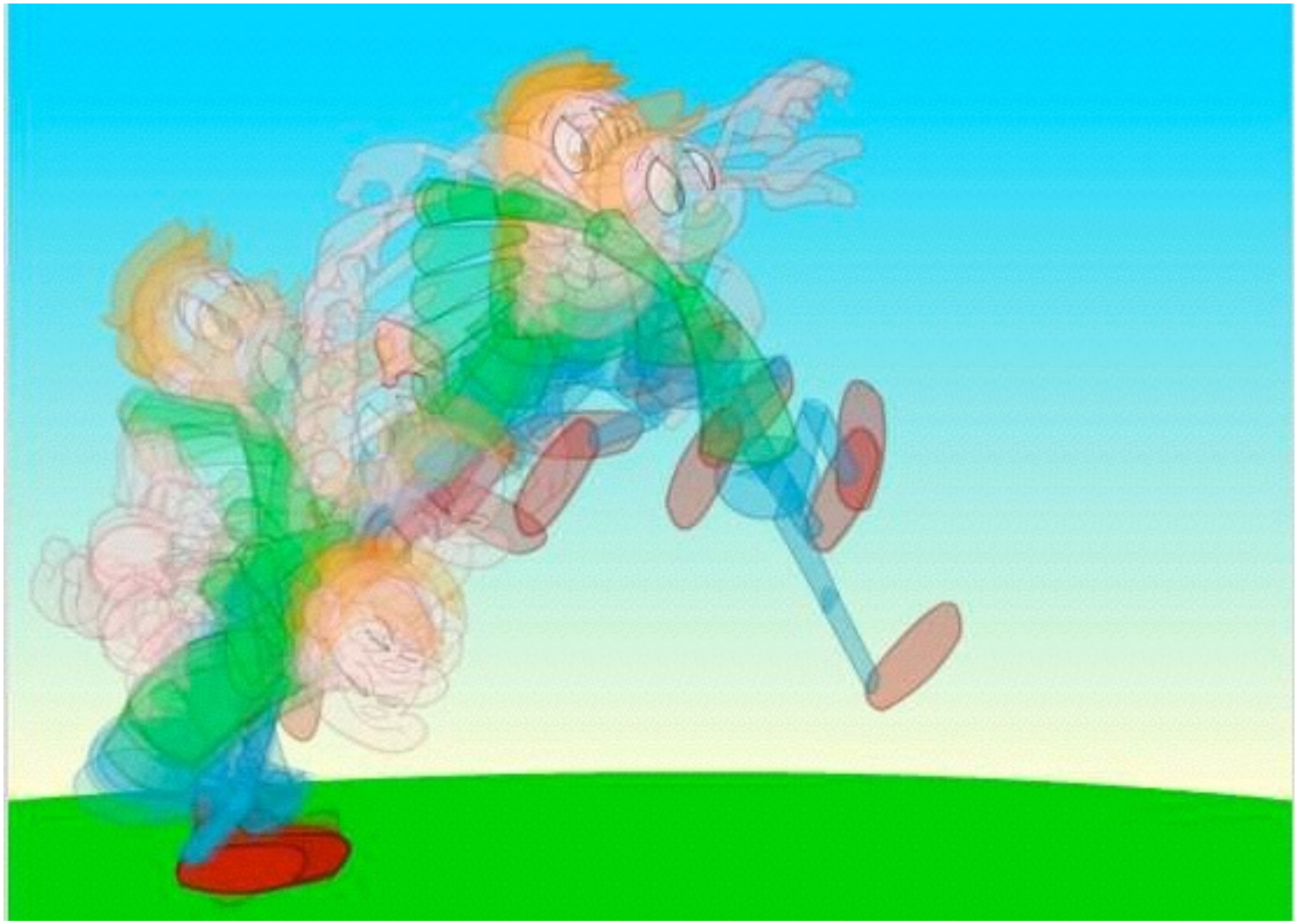




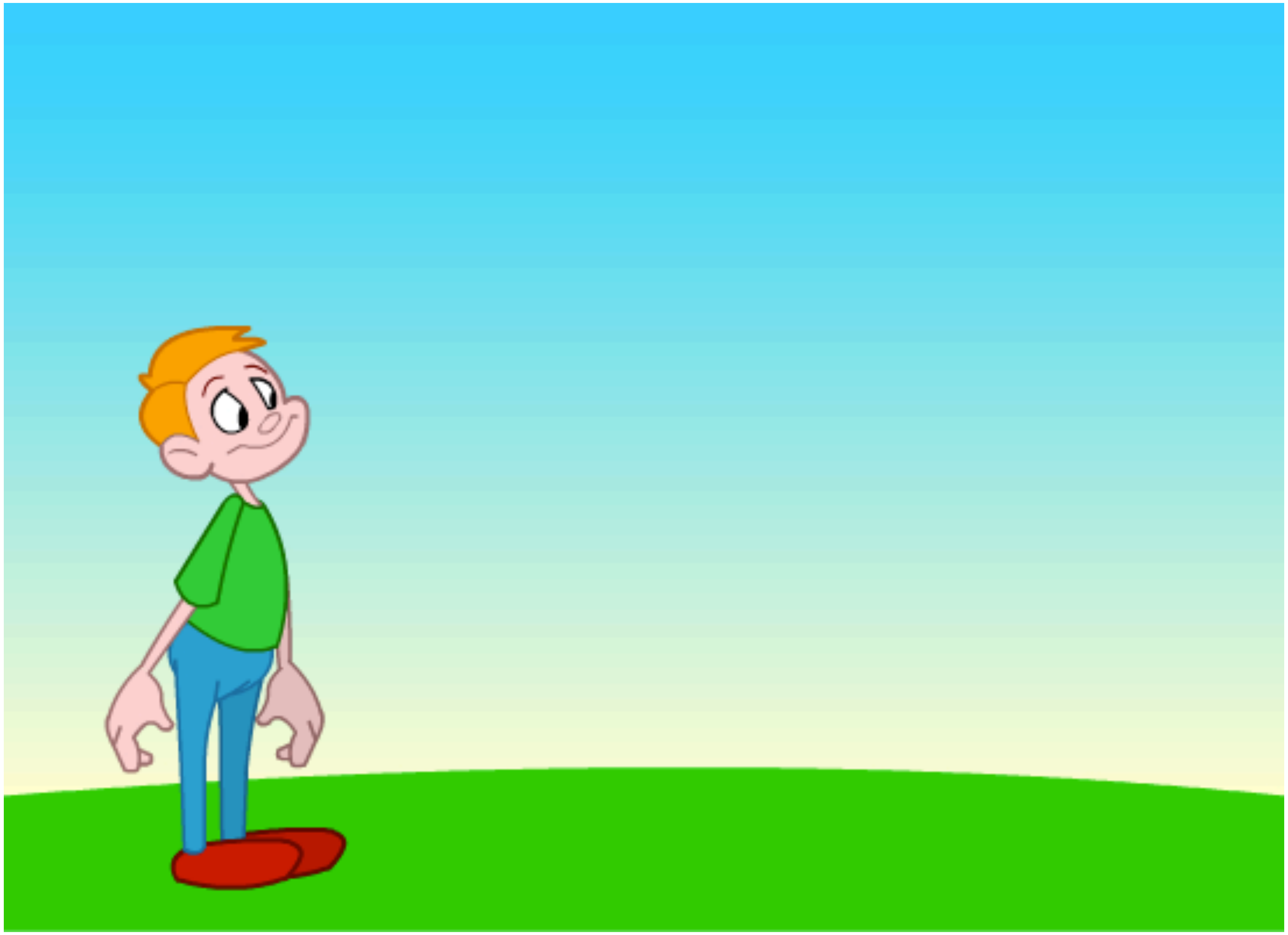
Anticipation/Reaction: Jumping Slime Ball



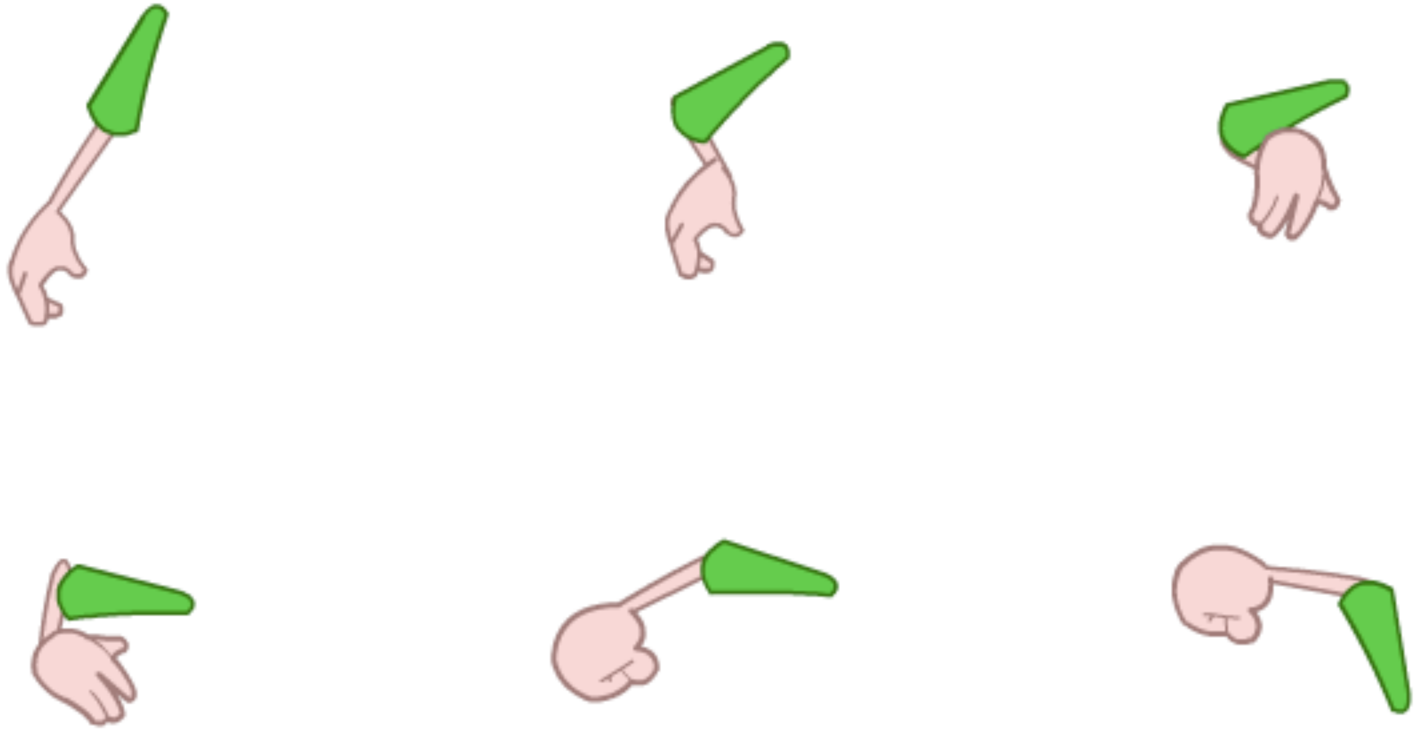
An Animated Comic Character (1)



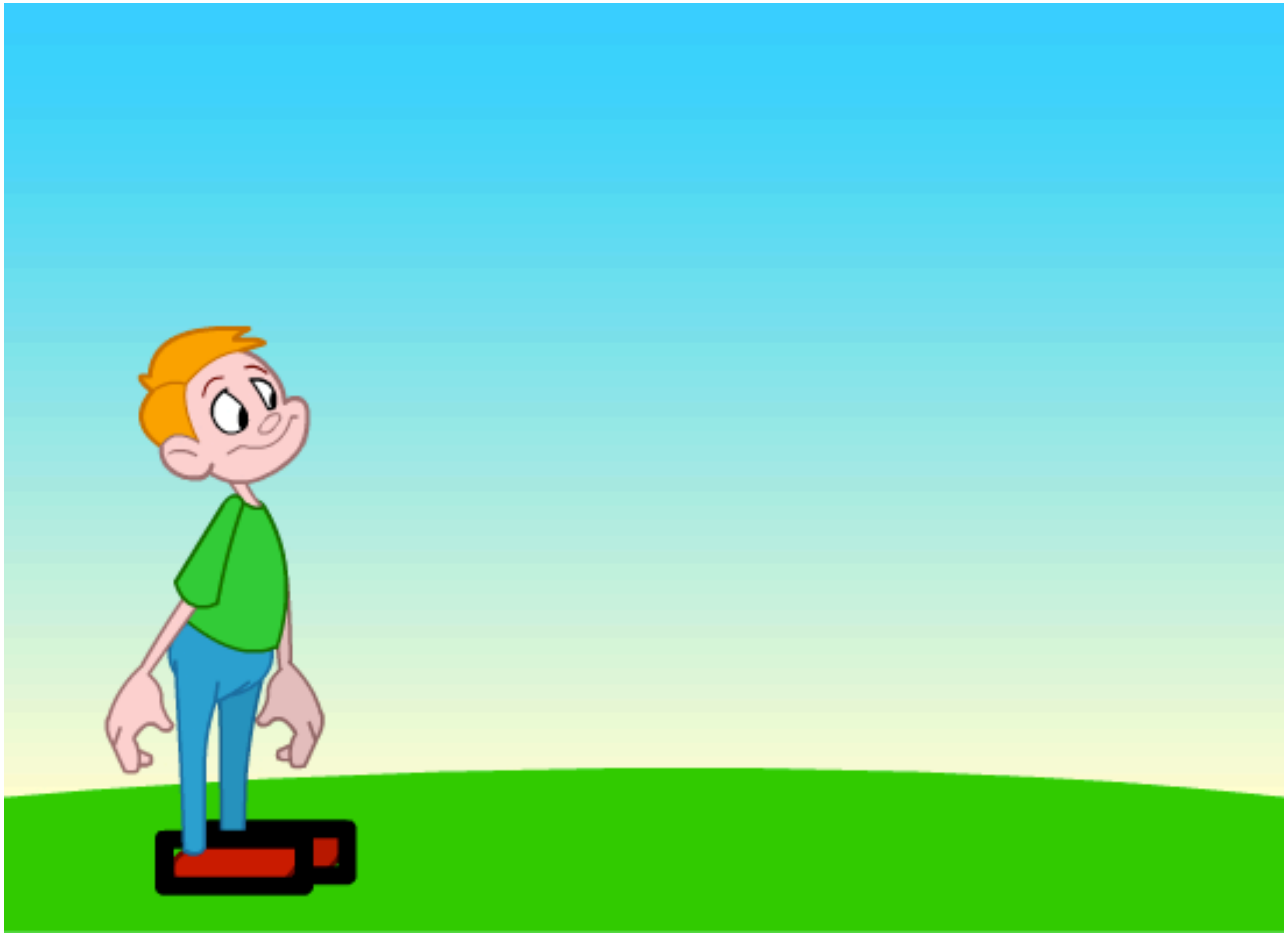
An Animated Comic Character (2)



Overlapping: Complexity of Movements



Smooth Arcs



The Story and Situation for a New Game

"The sound of the screaming alarms aboard the space cruiser abruptly awoke Space Kid, our ultimate hero of the futuristic universe, from his cryogenic nap. When our hero investigated, it was obvious his worst fears were now true. His loyal sidekick, teddy, had been bear-napped from the comfort of his own sleep chamber.

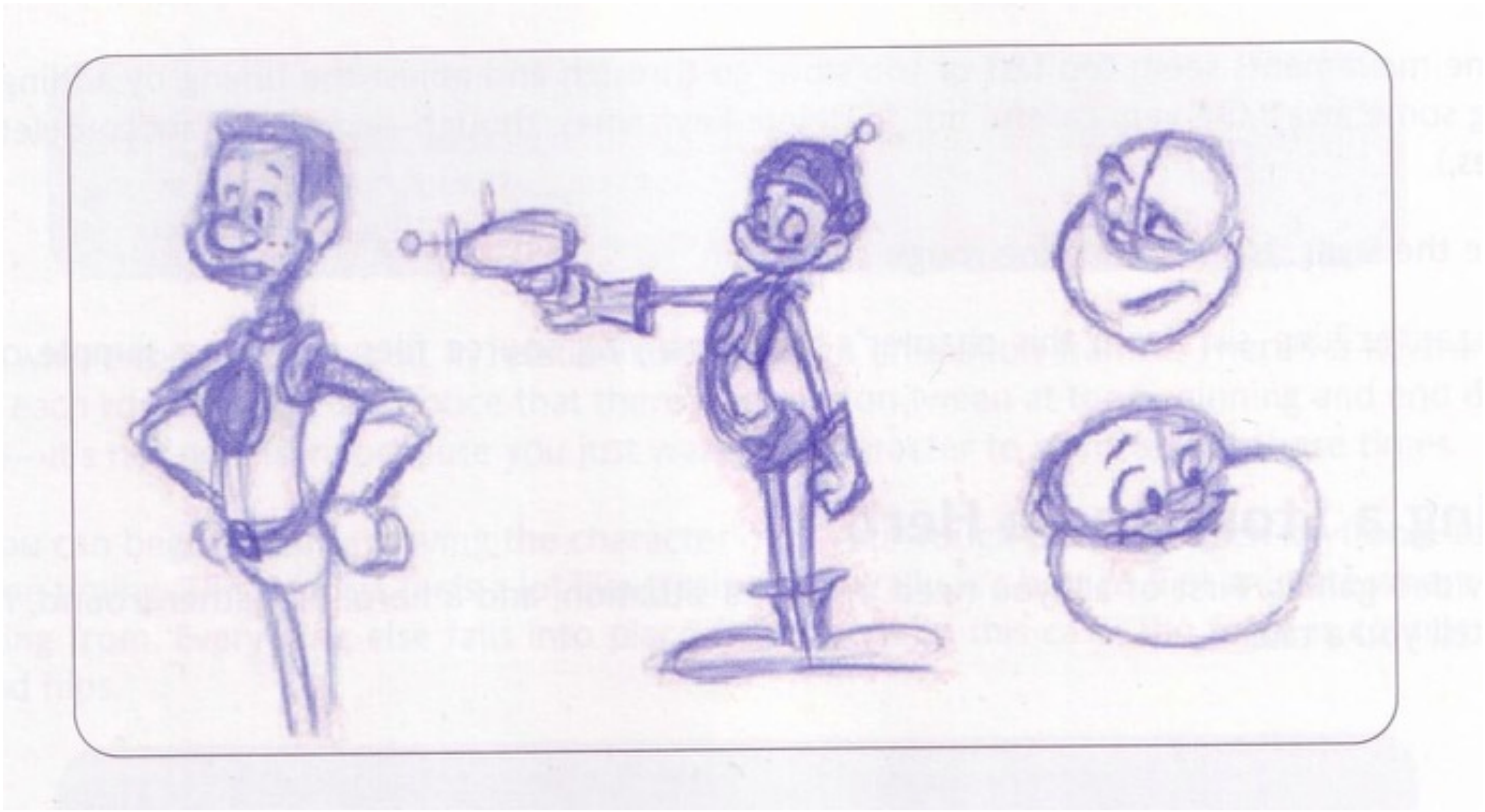
Immediately Space Kid knew there could only be one ruthless and vile enemy capable of committing such an atrocity: his longtime arch nemesis, Lord Notaniceguy.

Armed only with his trusty ray gun, Super Kid changes course for Quexxon Sector-G. Although our brave hero is fully aware he's falling for the bait in a trap, he must save Teddy."

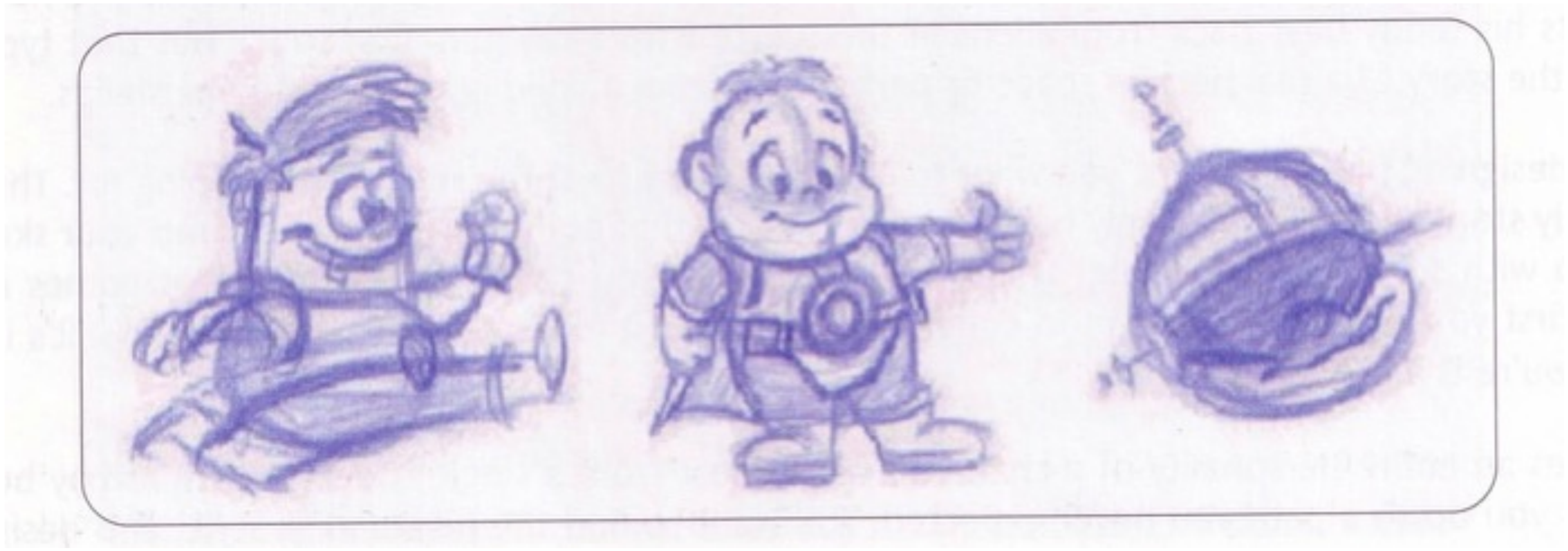
The Design Process

1. Create rough sketches of many different visual interpretations for the character (best with paper and pencil)
   – Brainstorming technique: Do *not* yet select!
2. Select among the gallery of characters according to compatibility with story, credibility, humour/seriousness, ...
3. Create rough sketches ***(paper and pencil)*** for the various animation sequences needed, e.g. run, jump, shoot, ...
   – Here usage of the authoring system can help already
4. Create optimized computer graphics for an “idle” animation.
5. Realize the animation sequences
   – Make sure that all sequences start and end with the idle position

Character Brainstorming (1)

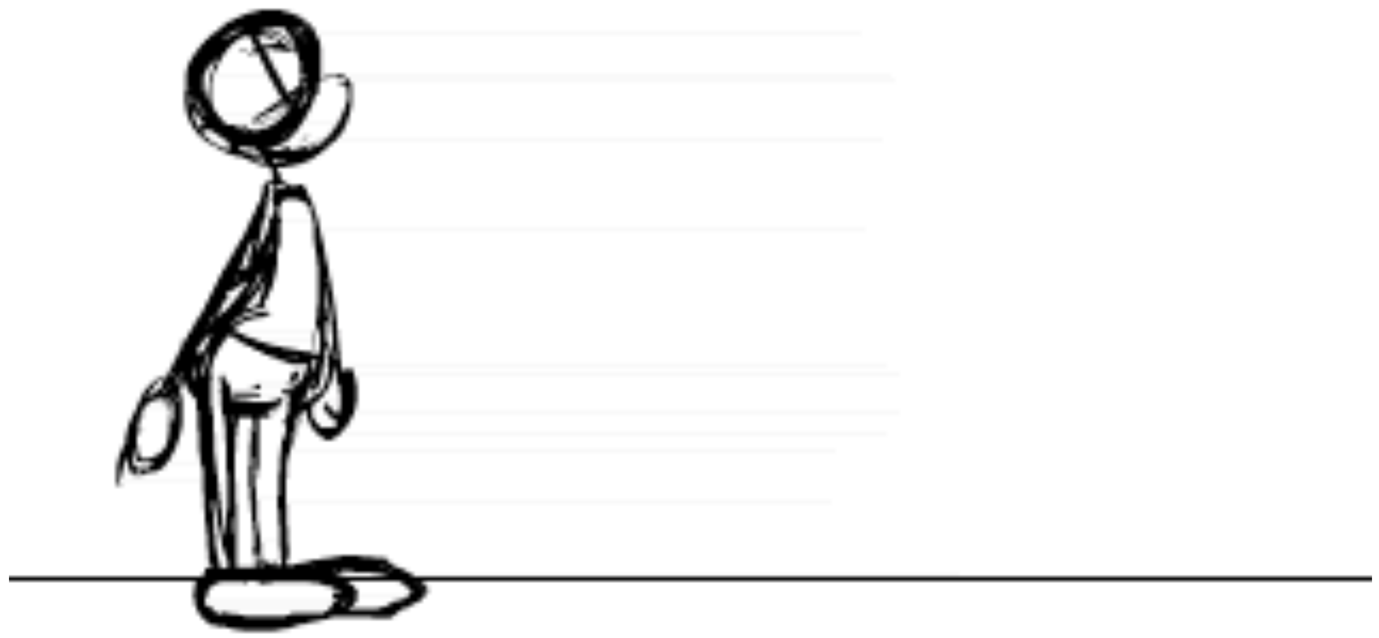


Character Brainstorming (2)



The Final Character

Rough Sketch for “Jump” Animation

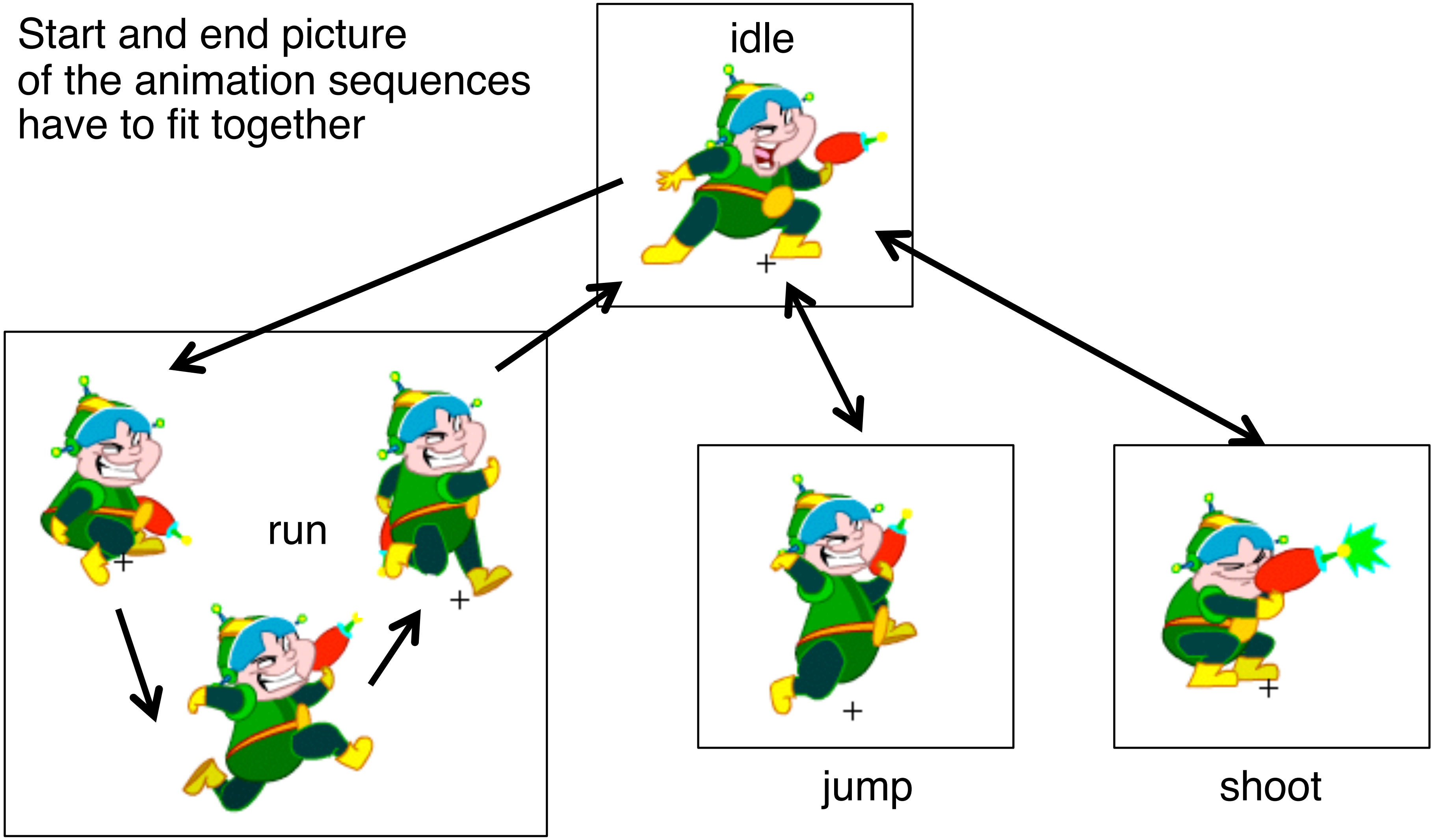


Physics and Animation

- Sprite animation:
  - Local changes in appearance
  - Often in a loop or a simple repetitive logic
  - May be parameterized
- Sprite movement:
  - Placement of sprite on stage, including speed and other parameters
  - Realized in program logic based on global event loop
- Consequence for movement animations (like jump):
  - Movements "as if staying on the ground"
  - Character design provides *central transition point* for the character
    - » In the middle of the bottom
    - » Must be the same point across all animation phases
    - » Used to determine whether ground has been hit, whether we are falling off an edge, ...

Continuity of Animation Sequences

Start and end picture
of the animation sequences
have to fit together



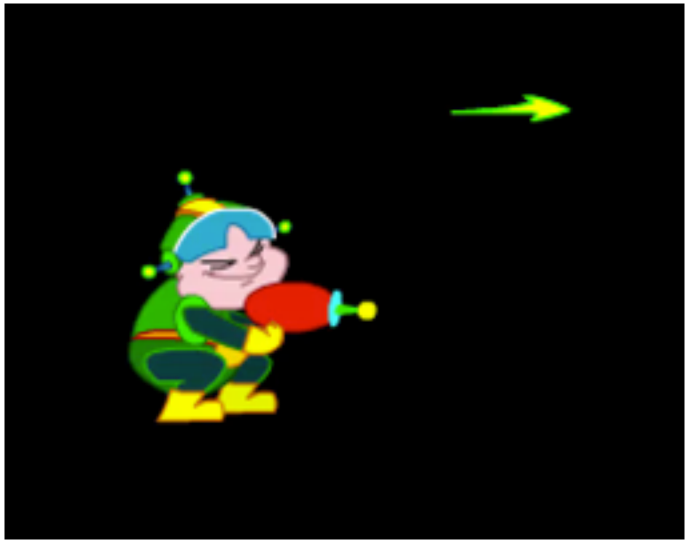
Example: State-Dependent Animations

idle

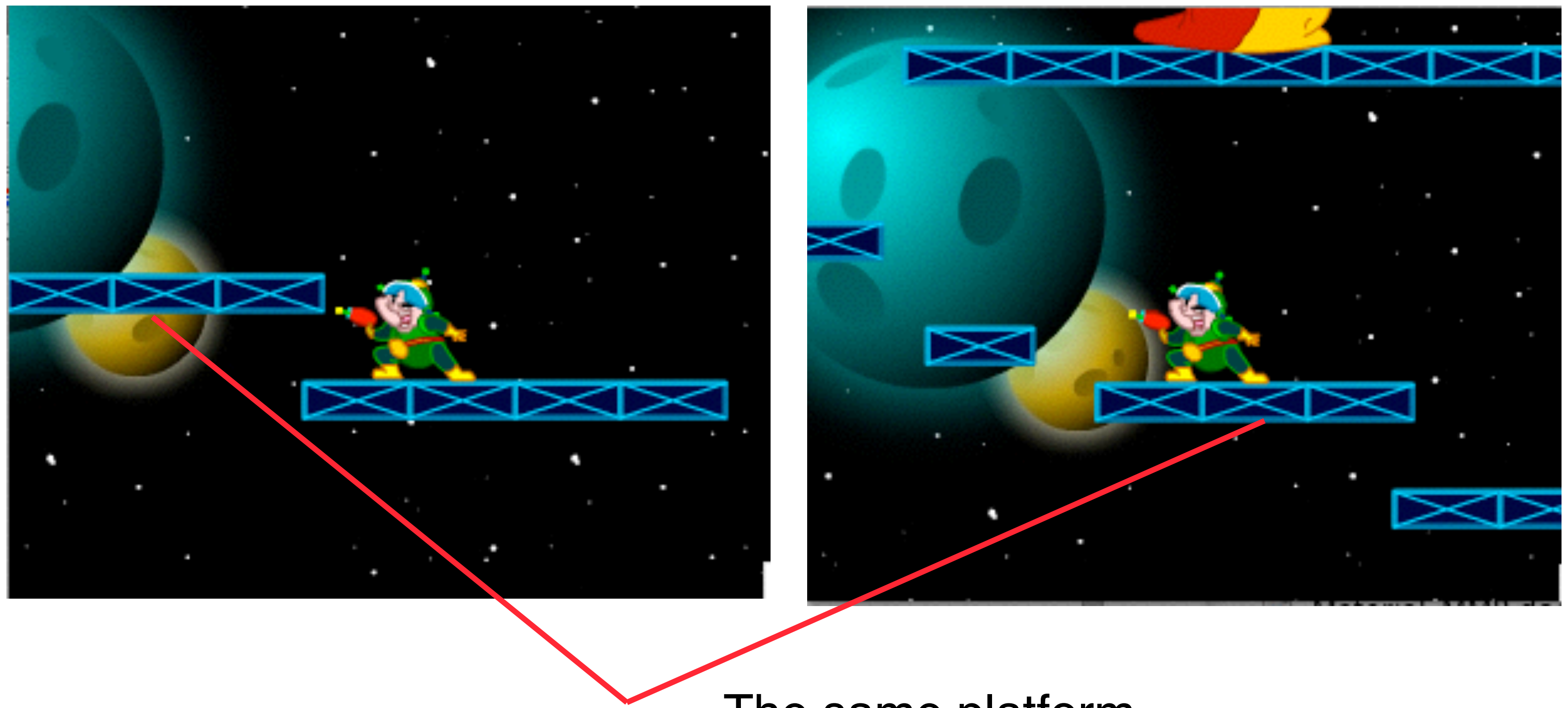
jump

run

shoot



Parallax/Multiplane Effect



The same platform