



DEPARTMENT OF
INFORMATION SYSTEMS
AND COMPUTER SCIENCE



```
00101110010001111011110010011010110100100101  
11010101101010100001010101010010101010101010  
10100101001001001010101010101010101010101010  
11100001111010110000000111101010101010000010101  
111010101110010100010010111010100010100100111010  
10101001010010010010000101010110101010101010010111  
00101010010101001010100000001010101001111101000011001  
100011001000011100110101011000100110101010000101010  
1100101010101000010011001010100010010101010101010  
10100101001001001010101010101010101010101010101010  
1110000111101011000000011110101010101010000010101  
0010010101001010010010100100010101010101010101010010  
1001010010000101010010010101001010010101010010010  
10010100101010101010010101001010010101001001001001  
10010100101010101010101010101010101010101010101010
```

										01
										02
										03
										04
										05
A	B	C	D	E	F	G	H			

Basic Physics

Aiming for Realism

Lecture Time!

- ▶ Position: Physics-based Movement
- ▶ Friction: Motion Isn't Perpetual
- ▶ Elasticity: Boing or Splat?

0010101001010100001111001101010010101
10001100100001111001101010010101
11001010101010100001001100101010100
100101001001001010101010101010101
11100001111010110000000111101001
0010010101001010010010100100101010
10010100100001010100100101001010
1001010010101001010010100101010101



DISCS

Position

- ▶ You already know how to set an object's position
- ▶ Which also means you already know how to change an object's position
 - ▶ In response to player input
 - ▶ Automated part of program loop

0010101001010100011110100001100
10001100100001111001101010010101
110010101010100001001100101010100
1001010010010010101010101010101
11100001111010110000000111101001
001001010100101001001010010010110
1001010010001010100100101001010
100101001010100101001010010101



Warning

- ▶ You have been exposed to enough code that I am now assuming that I don't have to explain each and every code detail:
 - ▶ Programming conventions (ex. a variable in all caps is a constant or some pre-defined value)
 - ▶ Undeclared variables (ex. copy-pasting slide code will require you to "fill in the blanks")



Some Adjustments

- Since we'll eventually be dealing with collisions, we need to modify our CircleShapes:

```
// assuming all CircleShapes  
// have the same radius  
anyCircle.setOrigin( RADIUS, RADIUS );  
anyCircle.setRadius( RADIUS );
```

Some Adjustments

- ▶ Position is already handled, but what about velocity and acceleration?
 - ▶ Each shape must have its own “current” velocity and “current” acceleration
 - ▶ How do we represent velocity?
 - ▶ How do we represent acceleration?

001010100101010001111001101010010101
10001100100001111001101010010101
110010101010100001001100101010100
100101001001001010101010101010101
11100001111010110000000111101001
001001010100101001001010010010110
1001010010001010100100101001010
100101001010100101001010010101



Time is Money

- ▶ You will also need a global constant representing your program's time step
 - ▶ Everything has to be scaled according to this value
 - ▶ By default, this should be equal to a frame's "time" in seconds

```
#define TIMESTEP      1.0f / FPS
#define FORCE          10000.0f * TIMESTEP
```



“Moving” an Object

- ▶ To move an object in real life:
 - ▶ Apply a force on the object
 - ▶ This causes the object to accelerate
 - ▶ Which changes the object's velocity
 - ▶ Which changes the object's position

0010101001010100001111001101010010101
110010101010100001001100101010100
100101001001001010101010101010101
11100001111010110000000111101001
001001010100101001001010010010110
1001010010001010100100101001010
100101001010100101001010010101

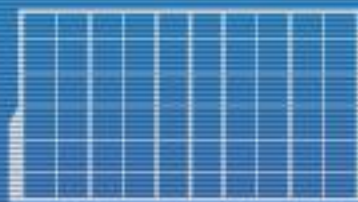


DISCS

Bare Minimum

- ▶ If we didn't care about realistic physics:
 - ▶ Check for anything changing acceleration
 - ▶ Determine current acceleration
 - ▶ Add acceleration to current velocity
 - ▶ Add current velocity to position
- ▶ Simple to understand and easy to implement, but not realistic
 - ▶ It's close, though!

001010100101010001111001101010010101
110010101010100001001100101010100
1001010010010010101010101010101
11100001111010110000000111101001
001001010100101001001010010010110
1001010010001010100100101001010
100101001010100101001010010101



Massive Force

- ▶ Force = Mass * Acceleration
 - ▶ How do we represent an object's mass?
 - ▶ To keep things simple, we will be using *rigid body* physics
 - ▶ Sort of like billiard balls
 - ▶ Any number of forces can be acting on an object
 - ▶ They can have different directions, too

001010100101010001111001101010010101
10001100100001111001101010010101
110010101010100001001100101010100
1001010010010010101010101010101
11100001111010110000000111101001
001001010100101001001010010010110
1001010010001010100100101001010
100101001010100101001010010101



Acceleration

- By default, an object is not accelerating
- Check for sum of all force vectors then compute for acceleration from there

```
// acceleration initialization here - what should go here?  
// process input for forces - but this is not quite correct?  
if( sf::Keyboard::isKeyPressed( keyRight ) )  
{  
    playerAccel.x += FORCE / MASS_PLAYER;  
}  
  
if( sf::Keyboard::isKeyPressed( keyUp ) )  
{  
    playerAccel.y -= FORCE / MASS_PLAYER;  
}
```

001010100101010001111001101010010101
11001010101010100001001100101010100
100101001001001010101010101010101
11100001111010110000000111101001
001001010100101001001010010010110
1001010010001010100100101001010
100101001010100101001010010101



DISCS

Velocity and Position

- After that, it's a simple matter of applying physics:

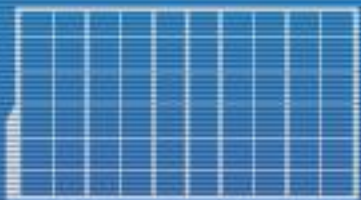
```
float t = TIMESTEP;
```

```
sf::Vector2f pos = circle.getPosition();
```

```
pos = ???;           // what should go here?
```

```
circleVel = ???;     // what should go here?
```

0010101001010100011110100001100
10001100100001111001101010010101
11001010101010100001001100101010100
100101001001001010101010101010101
11100001111010110000000111101001
001001010100101001001010010010110
10010100100001010100100101001010
100101001010100101001010010101



DISCS

Friction

- ▶ We can let friction cause an object to gradually lose velocity
- ▶ Friction is either on (1) or off (0)

```
// note: apply only if friction is enabled  
circleVel = ???;    // what should go here?
```

```
0010101001010100001111001101010010101  
10001100100001111001101010010101  
11001010101010100001001100101010100  
100101001001001010101010101010101  
11100001111010110000000111101001  
001001010100101001001010010010110  
1001010010001010100100101001010  
100101001010100101001010010101
```



Elasticity

- ▶ Some objects bounce, some don't
 - ▶ Particularly important if you have some kinematic surface acting as a floor or wall
- ▶ It depends on your elasticity coefficient
 - ▶ 0 = Splat
 - ▶ 1 = Boing
 - ~~▶ In-between = Sploing?~~

0010101001010100011110100001100
10001100100001111001101010010101
11001010101010100001001100101010100
100101001001001010101010101010101
11100001111010110000000111101001
001001010100101001001010010010110
1001010010001010100100101001010
1001010010101010010100101010101



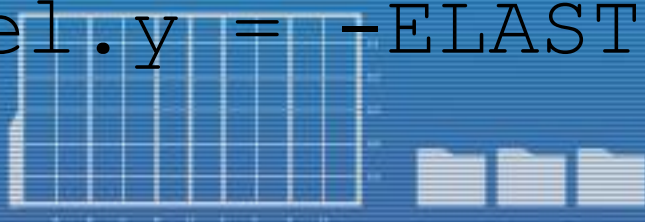
DISCS

Elasticity

- ▶ Assuming rigid body dynamics, you also have to account for penetration
- ▶ This means either adjusting the object's position upon impact or only applying the collision response if the object is indeed moving towards a surface

```
// assuming an object hitting the floor  
// note: this isn't treated like a force  
// (more on that when we get to impulse)
```

```
circleVel.y = -ELASTICITY * (circleVel.y);
```



Homework

- ▶ Modify your previous homework:
 - ▶ The only object active and visible should be ONE CircleShape
 - ▶ WASD keys should apply a corresponding directional force on that CircleShape
 - ▶ CircleShape should act as though it were an air hockey puck
 - ▶ The renderable area of your program window is the air hockey arena



Homework

- ▶ Modify your previous homework (continued):
 - ▶ You should be able to toggle friction on and off
 - ▶ Include a visual indicator (that isn't a console printout) so you can tell if friction is on or off without moving the puck
 - ▶ CircleShape collision with the edges of the program window should be totally elastic

► But I may ask you to adjust elasticity

