

# Currents: Coding with Cinder

Week 3: Object Oriented Programming / Particle System

Instructors

Luobin Wang ([luobin@newschool.edu](mailto:luobin@newschool.edu))

Weili Shi ([weili@newschool.edu](mailto:weili@newschool.edu))

# Bear with us!

Let's work together to beat this Cinder thing!

```
class Buddha {  
  ...  
};  
  
Buddha shakyaṃuni;
```

# Object Oriented Programming

In **OOP**, computer programs are designed by making them out of objects that interact with one another.

```
Buddha shakyauni;  
shakyauni.meditate(); // I don't know how Budda meditates but I just want him to.
```

# Data Encapsulation

Bundling the data, and the functions that use them

# Abstraction

Exposing only the interfaces, and hiding the implementation details from the user.

```
Buddha shakymuni(42); // Constructing a buddha with the ultimate answer...
```

# Constructors

A constructor is a special member function of a class that is executed whenever we create new objects of that class.

```
class Buddha : public Monk {  
    ...  
};
```

# Inheritance

Defining a class based on another class.

An opportunity to reuse the code functionality and fast implementation time.

Access	public	protected	private
Same class	yes	yes	yes
Derived classes	yes	yes	no
Outside classes	yes	no	no

# Access Control and Inheritance

# Polymorphism

A call to a member function will cause a different function to be executed depending on the type of object that invokes the function.



# ci::app::setFrameRate()

set your application frame rate, potentially making it more stable.

# gl::drawString()

Draw a string, easiest way to put text on. But you can't set font.

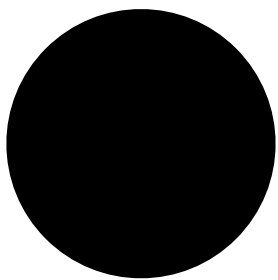
# Particles!

Not one particle.

But many of them!



# Particle



If you build a particle, what does it need

Velocity `velocity = velocity + acceleration`

Acceleration `acceleration += force`

Force `force is usually a vec2 (it needs direction)`

Friction `velocity *= friction`

force = m \* a

if m == 1 then force = a

Considering our particles are all of equal mass

`speed = speed + force`

Apply force to speed every frame

`force *= 0`

Force cannot be add up. so don't forget to reset it every frame

# speed \* = friction

We need friction, or particle will never stop

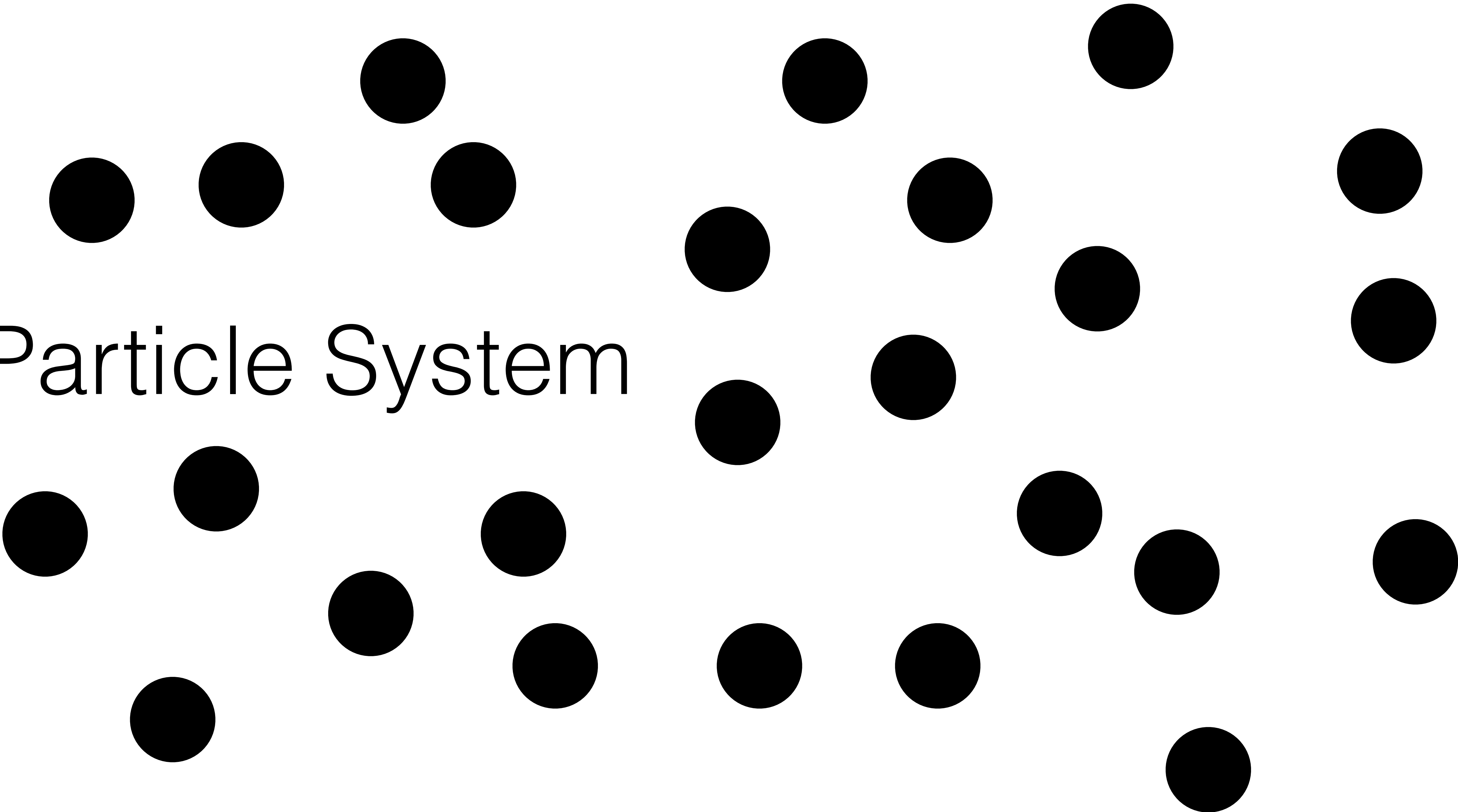


# ci::length(ci::vec2)

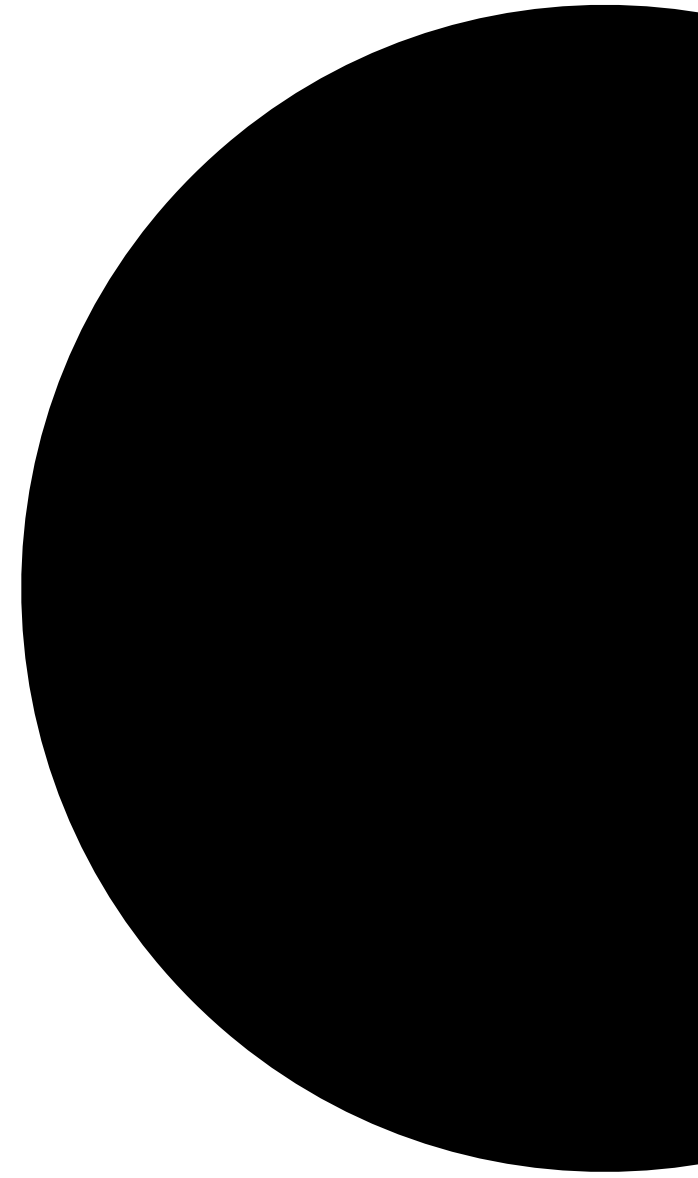
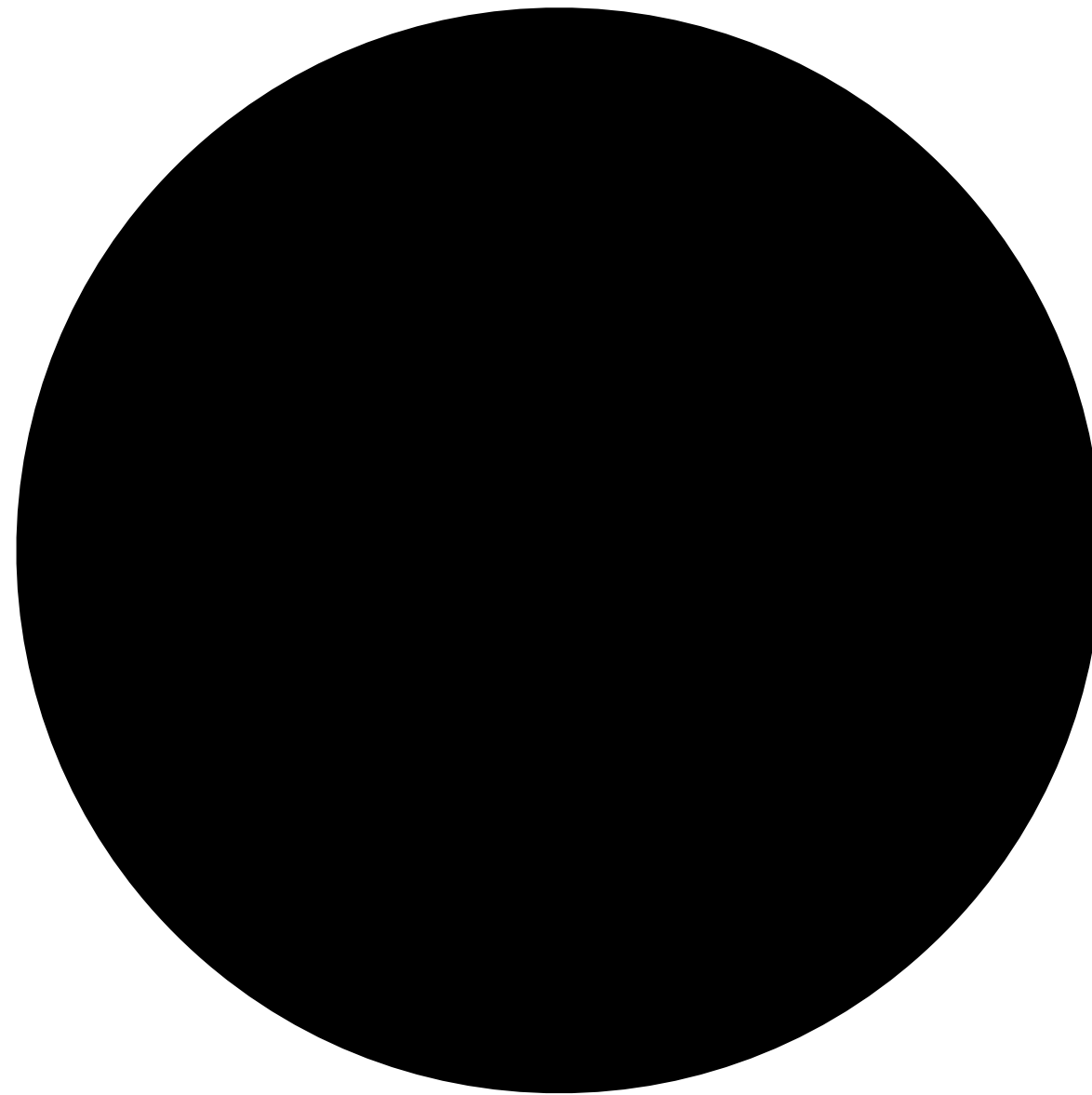
This function can get you the length of your vector

# Particle System ●

# Particle System

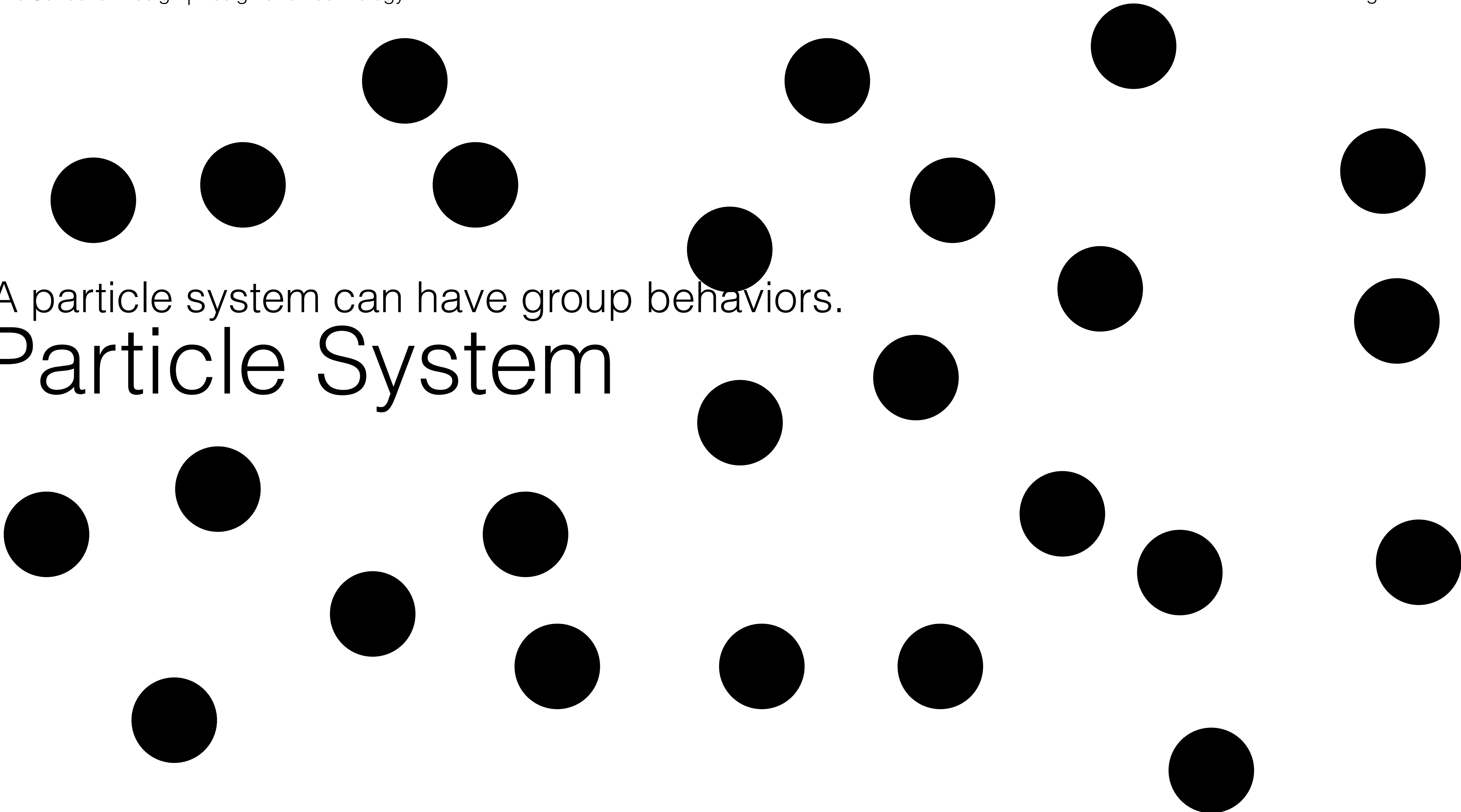


Individual particle will  
have their own  
parameters - like  
position, acceleration,  
and etc.



A particle system can have group behaviors.

# Particle System



# Navigating in console

List all files in directory:

ls

Go in to directory:

cd [directory]

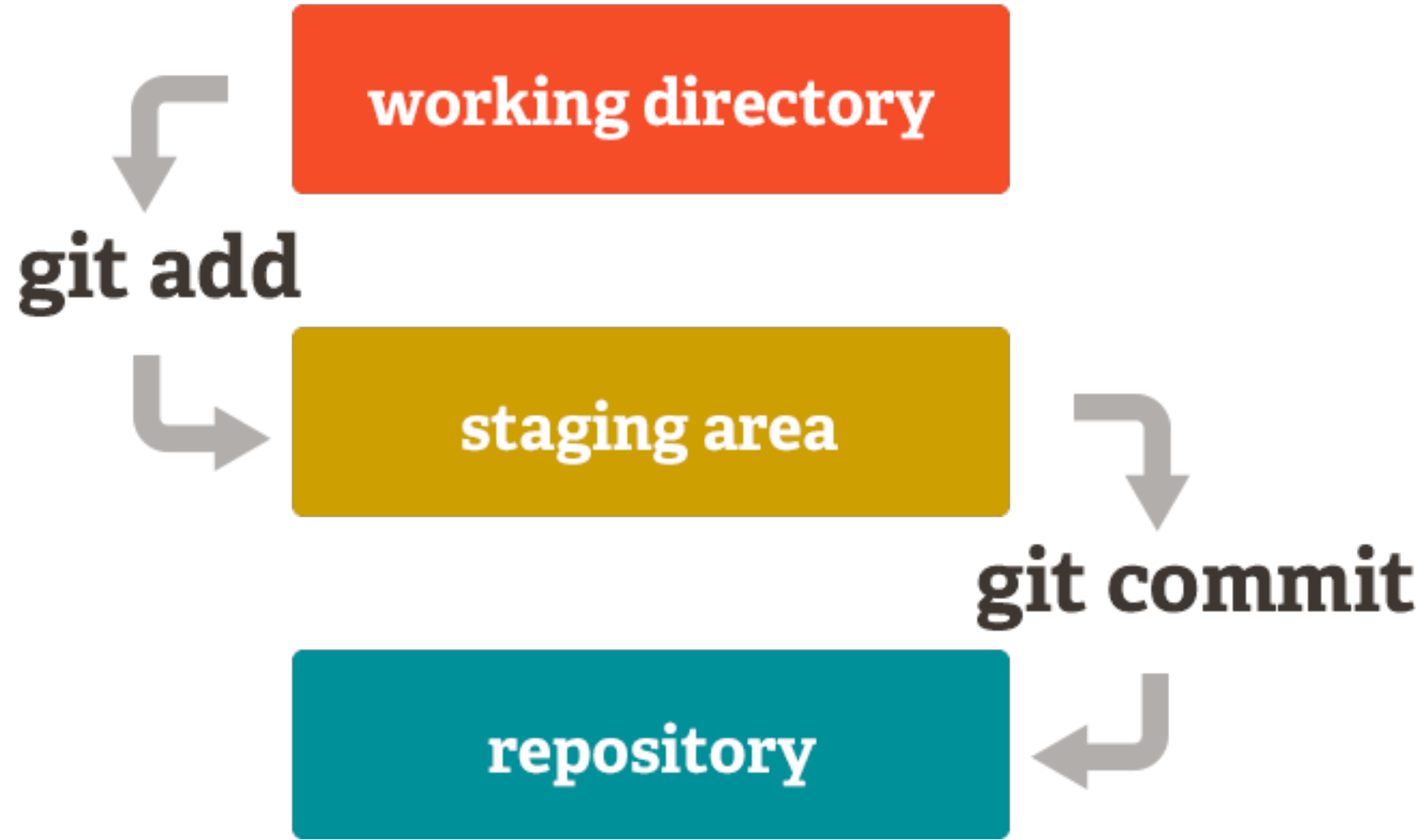
Go up:

cd ..

Open this folder:

open .

# Git is not GitHub





# Basic git commands

Initialize this directory:	<code>git init</code>
add file to HEAD:	<code>git add [file]</code>
commit current changes:	<code>git commit -m "your message"</code>
check your history:	<code>git log</code>
go to a particular history branch:	<code>git checkout [commit id]</code>

go here for a cheat-sheet, it is easy

<http://rogerdudler.github.io/git-guide/>

# Homework

1. Use a Particle System to simulate a phenomenon in nature (rain, snow)
2. (Bonus) Integrate images (ever wanna try png images?) into your particle system, so you can do something like snow flakes? or even smoke (a lot of games use image to simulate smoke).
3. (Bonus) Make multiple versions of your particle system which spawn different types of particle. Use inheritance to reduce duplicate code.

Due: Feb 14 (Tue).

Post a video demo to Slack channel. Keep your first 3 weeks' homework. We'll teach you how to upload it to GitHub next week.