**Instituto Politécnico Nacional**

**Escuela Superior de Cómputo**


# Evolutionary Computing

## Lab Session 7: Particle Systems

**Alumno: David Arturo Oaxaca Pérez**

**Grupo: 3CV11**

**Profesor: Jorge Luis Rosas Trigueros**

**Fecha de realización: 02/10/2021**

**Fecha de entrega: 04/10/2021**

## Marco teórico

Los sistemas de partículas tienen un origen curioso, pues fueron nombrados para los efectos especiales usados en Star Trek II por William T. Reeves en 1982 y de ahí se han vuelto bastante populares en el uso de animaciones, arte digital, video juegos y como modelo de varios tipos de fenómenos naturales irregulares, tales como el fuego, el humo, las cataratas, la niebla, las burbujas e incluso el pasto.

Éste término se define como una colección de varias partículas diminutas que representan un componente de un elemento en el sistema y que en conjunto representan un objeto difuso. Durante un período de tiempo éstas partículas son generadas dentro del sistema, se mueven e interactúan dentro del sistema hasta que finalmente mueren o cumplen alguna especie de objetivo determinado.

Muchos sistemas de partículas usan formas simples para las mismas, aplicando comportamientos y fenómenos muy básicos como puede ser la gravedad o las colisiones entre partículas, tomando en cuenta que cada partícula suele tener características como su propia velocidad, tamaño y posición dentro del sistema en el que interactúan.

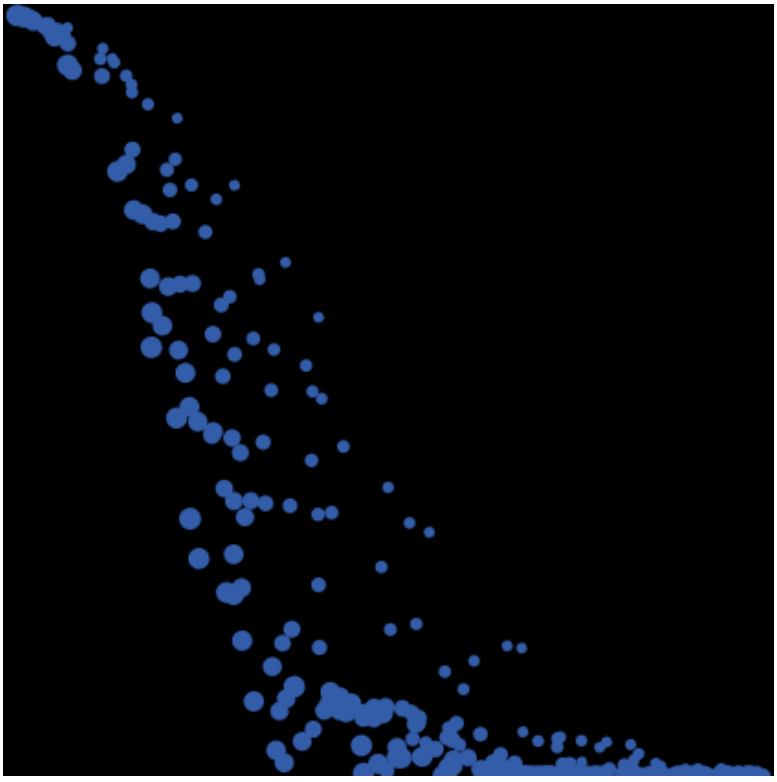

**Figura 1.** Sistema de partículas que simula la caída del agua.

A lo largo de la sesión veremos el uso de dos sistemas de partículas con condiciones especiales que requieren de la interacción con sus vecinos, como es las colisiones entre las partículas y la afectación de los cambios en su al rededor.

## Sistema de partículas con interacciones de sus vecinos más cercanos en una cuadrícula.

Lo que se requiere para éste sistema es que las partículas se encuentren ordenadas en una especie de cuadrícula, interactúen y cambien su comportamiento tomando en cuenta los cambios en sus vecinos más cercanos como fue visto en clase, pero esta vez de manera bidimensional en contraste con la línea de partículas.

De esta manera, lo que se pretende es que si una partícula cambia su posición de manera vertical hacia arriba, su vecino más cercano haga lo mismo emulando ese comportamiento de manera en que parezca una “ola” como las que suelen haber en partidos de soccer.

**Figura 2.** Una ola mexicana en un estadio.

## Sistema de partículas con colisiones.

Este sistema incorporó la física para calcular las colisiones que pudieran darse entre las partículas y la “pared”. Para esto fueron usadas las formulas vistas en clase que calculan el nivel de fuerza y la nueva velocidad que adquiere una partícula tras un choque. Para el cálculo de la fuerza se usaron las siguientes ecuaciones:

$$\overrightarrow{F_{12}} = \{|\overrightarrow{r_{12}}| > 2,\ \ \vec{0} = (0,0)\ \ \ |\overrightarrow{r_{12}}| \leq 2,\ \ \vec{0} = \frac{r_{12}}{|\overrightarrow{r_{12}}|^2}$$

Donde $\overrightarrow{r_{12}}$ viene siendo la distancia entre dos partículas. Bajo estas circunstancias, si es menor a dos implica que estas están colisionando

Las ecuaciones para esta posición son las siguientes:

$$\overrightarrow{r_{12}} = (x_2 - x_1\ , x_1 - \ x_1)$$

$$r_{12} = |\overrightarrow{r_{12}}| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Para el cálculo de la velocidad, simplemente lo despejamos usando algunas fórmulas para el cálculo de la fuerza y la velocidad. La ecuación para la velocidad y la forma en la que se obtiene es la siguiente:

$$\overrightarrow{F} = m\overrightarrow{a} \quad \rightarrow \overrightarrow{a} = \frac{\overrightarrow{F}}{m} = \overrightarrow{F}$$ - Si consideramos que el peso es 1.

$$\overrightarrow{V_{t+1}} = \overrightarrow{V_t} + \overrightarrow{a} = \overrightarrow{V_t} + \overrightarrow{F}$$

Usando las ecuaciones anteriormente descritas se realizará un sistema donde las partículas pueden colisionar, teniendo como objetivo salir de un recuadro donde se encuentran "atrapadas", esto se realizará con dos variaciones:

Una sin un obstáculo cerca de la salida y otra con un obstáculo, esto para observar si el posicionamiento de un obstáculo genuinamente tiene un cambio en como las partículas logran escapar del recuadro donde se encuentran.

## Material y equipo.

- El programa fue realizado en Google Colaboratory usando Python 3.9 como lenguaje.
- El sistema que se utilizó para acceder a Google Colaboratory fue una laptop de marca Lenovo, modelo Ideapad S540 con Ryzen 7 y 8 GB de RAM, con sistema operativo Windows 10.

## Desarrollo.

### Sistema de partículas con interacciones de sus vecinos más cercanos en una cuadrícula.

Para el primer sistema de partículas requerido en esta sesión de laboratorio, las partículas deben actuar considerando las acciones de su vecino más cercano, para lograr simular una "ola mexicana", donde una persona se mueve y las otras continúan con este movimiento durante algún partido.

El primer paso para esto, fue analizar el código visto en clase que realiza una función similar con una sola hilera y a partir de este, cambiar el ciclo donde se grafican las partículas para que esta vez hiciera varias filas. Para hacer esto modificamos algunas de las variables globales existentes en el programa, cambiamos N a 20, donde N será el número de partículas por fila y columna y posteriormente modificamos la declaración de las variables "particles" y "up" para que sean inicializadas como matrices de 0 de NxN. También modificaremos la variable "r" que será el radio inicial de las partículas, en este caso será de tamaño 2. Cabe aclarar que cada elemento de la variable "particles" representara los cambios de una

partícula de nuestro sistema y cada elemento de la variable “up” será un valor binario, es decir 0 o 1, que representara si una partícula esta “levantada” en la ola.

```
N=20
particles = np.zeros((N,N))
up= np.ones((N,N))

wIm = ipw.Image()
display.display(wIm)


maxX=500
maxY=500
x0=100
y0=100
r=2
```

**Figura 3.** Modificaciones realizadas en las variables globales para representar la “Mexican Wave” en una cuadrícula de manera bidimensional.

Después de eso modificaremos la función “graph_particles” pues en esta es en donde se grafican las partículas de nuestro sistema, no es de mucha utilidad tener una representación bidimensional de estas partículas si no podemos visualizar ningún cambio, así que cambiamos el ciclo donde se grafican las partículas por dos ciclos (uno anidado dentro de otro), para representar las N filas con N partículas en cada una.

Otro cambio necesario en esta función fue la modificación del movimiento de las partículas al “levantarse”. Originalmente al estar en una hilera, éstas incrementaban su posición en el eje vertical, pero en éste nuevo sistema donde se están manejando como una cuadricula bidimensional, se ve un poco extraño y no representa el efecto deseado, así que se optó por mejor incrementar su radio; de manera que desde una perspectiva “aérea” se vea como si se estuvieran levantando para hacer una ola. Para esto se le sumará al radio de cada partícula su correspondiente elemento en la matriz “particles” previamente declarada, donde se almacenará el cambio de estado de la partícula (que en este caso será un aumento de radio).

```
def graph_particles(img):
  global particles,x0,y0,maxX,maxY,N, r
  stridex = (maxX - 2*x0)/N
  stridey = (maxY - 2*y0)/N
  for i in range(N):
    for j in range(N):
      cv2.circle(img,(int(x0+stridex*i),int(y0+stridey*j)),int(r+particles[i][j]),(255,255,255),-1)
```

**Figura 4.** Modificaciones en la función “graph_particles” para graficar una cuadrícula de partículas e incrementar los radios de las partículas para representar el efecto de una “Mexican wave” posteriormente.

El siguiente cambio fue en la función de “iteration”, que es la que realizara los cambios en las partículas en el tamaño de sus radios, para que cuando se llame posteriormente a la función “graph_particles” para graficar nuevamente a la cuadrícula de partículas, se vean representados sus cambios en la nueva iteración del sistema.

Para esto cambiamos los vecinos de los que obtiene su influencia, ya que no solo es el vecino izquierdo y el derecho (para esta forma particular de ola consideraremos que una partícula obtendrá su influencia de las partículas que se encuentran en diagonal a ella) . Estas contribuirán sumando el equivalente al 10% de su cambio de radio para el cambio de radio de la partícula vecina. También se dan las condiciones que determinaran si una partícula sigue “levantada” o no, que en este caso será el tamaño de su radio. Si este cambio de radio es mayor o igual a 5 se detendrá, mientras que si tiene un cambio menor a 0.5 se volverá a continuar.

Finalmente la función se queda con la misma instrucción que el programa original donde el cambio de las partículas tendrá un decaimiento del 10% cada iteración.

```
def iteration():
  global particles, N
  #influence of neighbors
  for i in range(N):
    for j in range(N):
      if i>0 and i<N-1 and j>0 and j<N-1 and up[i][j]==1:
        # Inicio de ola desde el centro
        # particles[i][j]+= 0.1*particles[i][j-1]
        # particles[i][j]+= 0.1*particles[i-1][j] + 0.1*particles[i+1][j]
        # particles[i][j]+= 0.1*particles[i][j+1]

        # Inicio de ola desde la primera fila
        particles[i][j]+= 0.1*particles[i-1][j-1] + 0.1*particles[i+1][j-1]
        particles[i][j]+= 0.1*particles[i-1][j+1] + 0.1*particles[i+1][j+1]
      if up[i][j]==1 and particles[i][j]>=5:
        up[i][j]=0
      if up[i][j]==0 and particles[i][j]<=0.5:
        up[i][j]=1
      #decay
      particles[i][j]*=0.9
```

**Figura 5.** Modificaciones realizadas en la función “iteration” para cambiar la influencia de los vecinos y las condiciones que harán que estos sigan teniendo un efecto.

El último cambio que realizamos para este sistema de partículas es definir que partículas iniciaran con la ola. Si únicamente una iniciara la ola, ésta tendría un efecto de que va en diagonal - para ser más apegados a la imagen proporcionada en clase- esta ola se iniciara en la segunda fila, ya que se considera que las primeras y últimas filas y columnas son “paredes”. En este caso las partículas que iniciarán la ola serán las que tengan como índice un número para que pertenezcan a la segunda fila (Los índices en el llamado a la variable se verían como [2][1], [4][1], …, [18][1]), estas iniciaran con un cambio de 1.5.

```
# Particulas que iniciaran con la ola
for i in range(9):
  particles[(i+1)*2][1]=1.5
```

**Figura 6.** Modificaciones en las partículas que contaran con un cambio inicial para comenzar la ola.

Tras esos cambios podemos visualizar como tenemos el efecto de una ola en dos dimensiones, pues las partículas van “levantándose” siguiendo la influencia de sus vecinos.

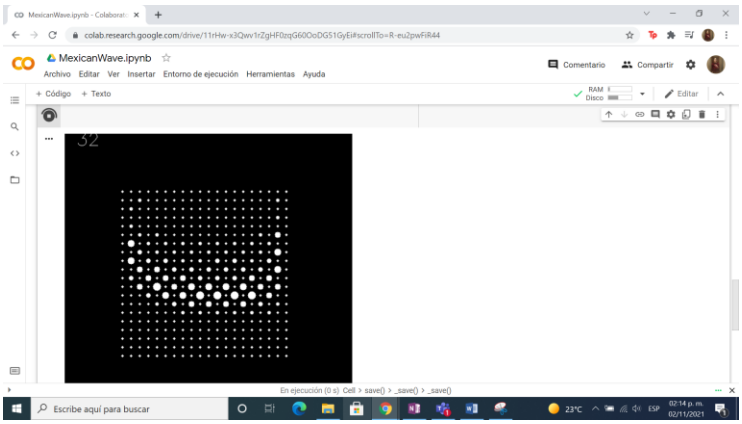


**Figura 7.** Captura de pantalla de una “Mexican wave” en una cuadrícula para un sistema de partículas en 2D.

A veces, incluso se puede ver un efecto más acumulativo en la ola, como se puede ver en la siguiente pantalla, aunque este decrece posteriormente.

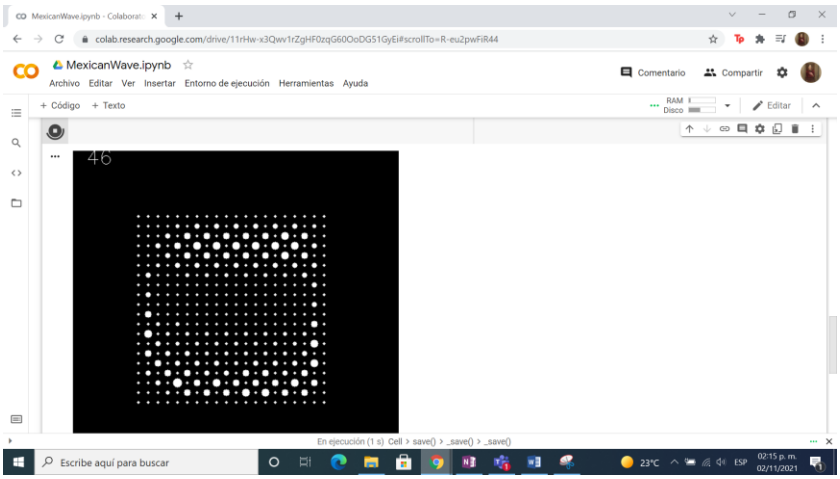


**Figura 8.** Otro ejemplo del sistema de partículas en donde existe influencia de sus vecinos.

Otro ejemplo de este sistema es hacer una ola que inicia desde el centro. Para esto se realizan ciertos cambios que aparecen comentados en las modificaciones previas y la partícula que inicia la ola, es solo una y es la que se encuentra en el centro de la cuadrícula. La forma en la que se ve ésta ola es la siguiente; parece que forma un circulo, como cuando cae una gota de agua a otro cuerpo de agua y crea ondulaciones en este último.

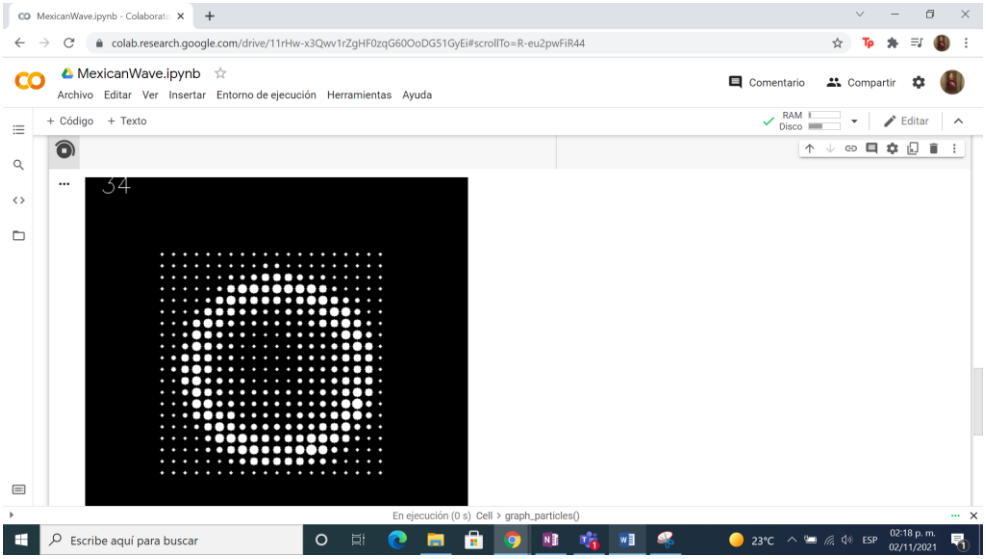


**Figura 9.** Una ola iniciada desde el centro con una sola partícula dando la perspectiva de ser una ondulación.

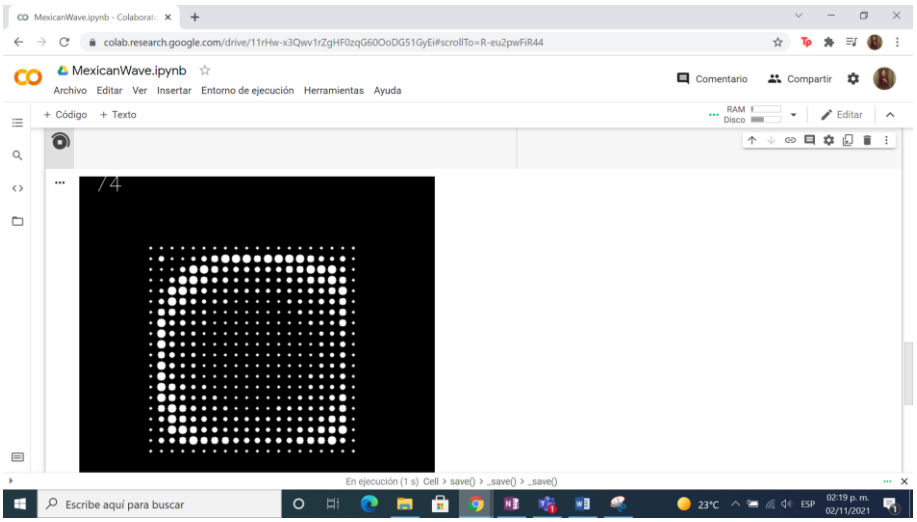


**Figura 10.** Otra captura pantalla de la ola llegando a los límites del sistema y dispersándose.

## Sistema de partículas con colisiones.

Para el sistema de partículas con colisiones primero tomamos en cuenta ciertas consideraciones:

En primer lugar, se deben de realizar dos casos, uno donde existe una pequeña venta de escape entre las paredes hacia la izquierda y el segundo donde sucede esto mismo, pero se agrega un obstáculo junto a esta ventana para ayudar a dispersar a las partículas.

El primer cambio que realizamos fue tomar en cuenta un error que hacía que las partículas atravesaran las paredes en el programa visto en clase. Las partículas tienen un radio de 10 para que sean visibles y se vean claramente, pero las ecuaciones implementadas para el cálculo de la fuerza solo consideran un radio de 1, haciendo que no se detecten bien las colisiones a menos que dos partículas choquen directamente en sus centros. Es por ello que el primer cambio se realizó en la función “calculateF” de la clase “Particle”, donde consideramos la siguiente figura para determinar la colisión entre dos partículas.

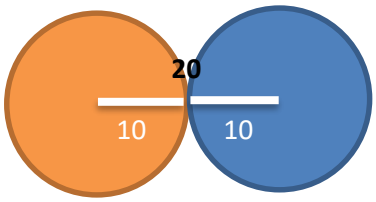

**Figura 11.** Representación de la distancia mínima en una colisión entre dos partículas con radio un radio de 10 unidades.

Por lo que, sí existe una distancia menor o igual a 20 entre 2 partículas consideraremos que existe una colisión entre ellas. Este cambio se verá representado en la siguiente figura.

```
def calculateF(self,r2):

  if self.WallParticle==True:
    return np.array([0.,0.])
  r12=self.r-r2
  r12magnitude=np.linalg.norm(r12)
  if r12magnitude<=20:
    #print(r12magnitude)
    return self.normalize_vector(r12)/ (r12magnitude**2)
  return np.array([0.,0.])
```

**Figura 12.** Modificaciones realizadas en la función “calculateF” para el funcionamiento de las colisiones para partículas con radio 10.

El siguiente cambio fue hacer un recuadro más amplio, haciendo las paredes más grandes, eliminar una pequeña parte de la pared derecha que sirviera como salida para las partículas y en el caso del segundo sistema, construir un obstáculo cerca de la salida.

```
# Paredes del recuadro donde inician las particulas
lineOfWallParticles(-200,150,200,150,45)
lineOfWallParticles(-200,-150,200,-150,45)
lineOfWallParticles(-200,150,-200,-150,35)

# Paredes que conforman la pared donde existe una ventana de salida
lineOfWallParticles(200,-150,200,-30,20)
lineOfWallParticles(200,150,200,30,20)

# Paredes del obstaculo
lineOfWallParticles(90,45 ,120, 0,20)
lineOfWallParticles(120,0 ,90,-45,20)
lineOfWallParticles(90,45 ,60, 0,20)
lineOfWallParticles(60,0 ,90,-45,20)
```

**Figura 13.** Paredes de partículas que forman parte del sistema como obstáculos o como una pared del recuadro donde inician las partículas.

Otro cambio que no fue tan significativo pero que ayudó a diferenciar a cada una de las partículas que no forman parte de la pared, fue la adición de un atributo para el

color que es pasado a la clase “Particle”, en este caso al crear las partículas dentro del sistema que no conformaran parte de una pared pasaremos un color aleatorio.

```
for i in range(40):
  random_color = (random.randrange(30, 255),random.randrange(30, 255),random.randrange(30, 255))
  particles.append(Particle(-1*random.random()*180, random.uniform(-1, 1)*120, 0.5, random.uniform(-0.8, 0.8), random_color))
```

**Figura 14.** Creación de las partículas del sistema con la inclusión de un parámetro que le dará el color a la partícula de manera aleatoria.

Lo siguiente que se realizó fue la aplicación de un par de condiciones que le otorgaran a las partículas del sistema la capacidad de ir hacia la izquierda y facilitaran el proceso de salida; adicional a eso se aplicó otra condición para un caso algo particular que fue encontrado en una de las iteraciones, cuando una partícula se encontraba en plena colisión con una pared, si otra llegaba y chocaba contra ésta, la partícula que se encontraba colisionando adquiría un mayor empuje y por lo tanto la fuerza que ejercía la pared ya no era suficiente por lo que muchas veces ésta lograba escapar.

Tomando en cuenta lo  mencionado previamente, se modificó la función “update_v” de la clase “Particle” para que si una partícula está avanzando hacia al lado contrario de la salida después de cierto punto, cambie su trayectoria hacia la izquierda y si la partícula está por atravesar la pared, cambie su dirección aún si cuenta con el empuje de otra partícula. Estos cambios se pueden ver reflejados en la siguiente figura:

```
def update_v(self):
  if self.WallParticle==True:
    return
  self.v+=self.F
  if self.r[0] < 0 and self.v[0] < 0:
    print("Aplica cambio de direccion")
    self.v[0] = 0.9

  if self.r[1] < -150 or self.r[1] > 150  :
    print("Aplica cambio de direccion")
    self.v = -1*self.v

  vmag=np.linalg.norm(self.v)

  if vmag> self.MaxV:
    self.v=self.normalize_vector(self.v)*self.MaxV

  return
```

**Figura 15.** Modificaciones realizadas a la función update_v para la consideración de ciertos casos que inciten a la partícula a ir hacia la izquierda.

Tras esas modificaciones podemos ver que obtenemos los siguientes resultados:

En la siguiente captura podemos ver el resultado de partículas que tienen la tendencia de avanzar hacia la izquierda tratando de salir mediante una pequeña ventana.

Un problema que podemos observar en éste caso es que cuando se aglomeran muchas en la salida, empiezan a chocar unas con otras, dificultando el proceso de evacuación. Otra situación que dificulta esto, es que las partículas únicamente dependen de las colisiones entre ellas para adquirir la dirección correcta hacia la salida, lo que igual la dificulta.

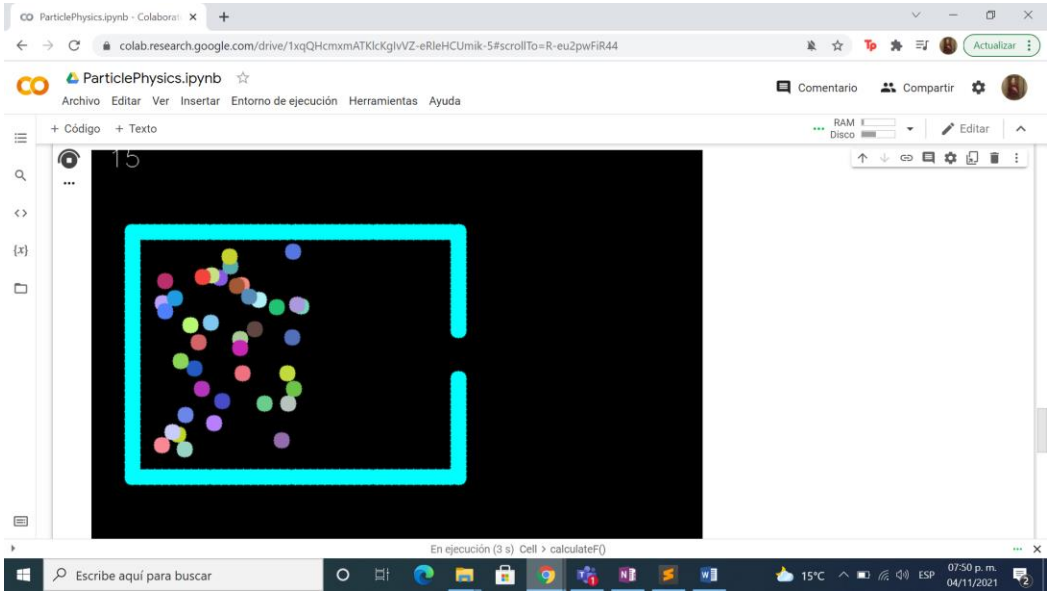


**Figura 16.** Captura de pantalla de un sistema de partículas con colisiones y una ventana de salida sin obstáculos en el camino en un inicio con 40 partículas dentro de las paredes.

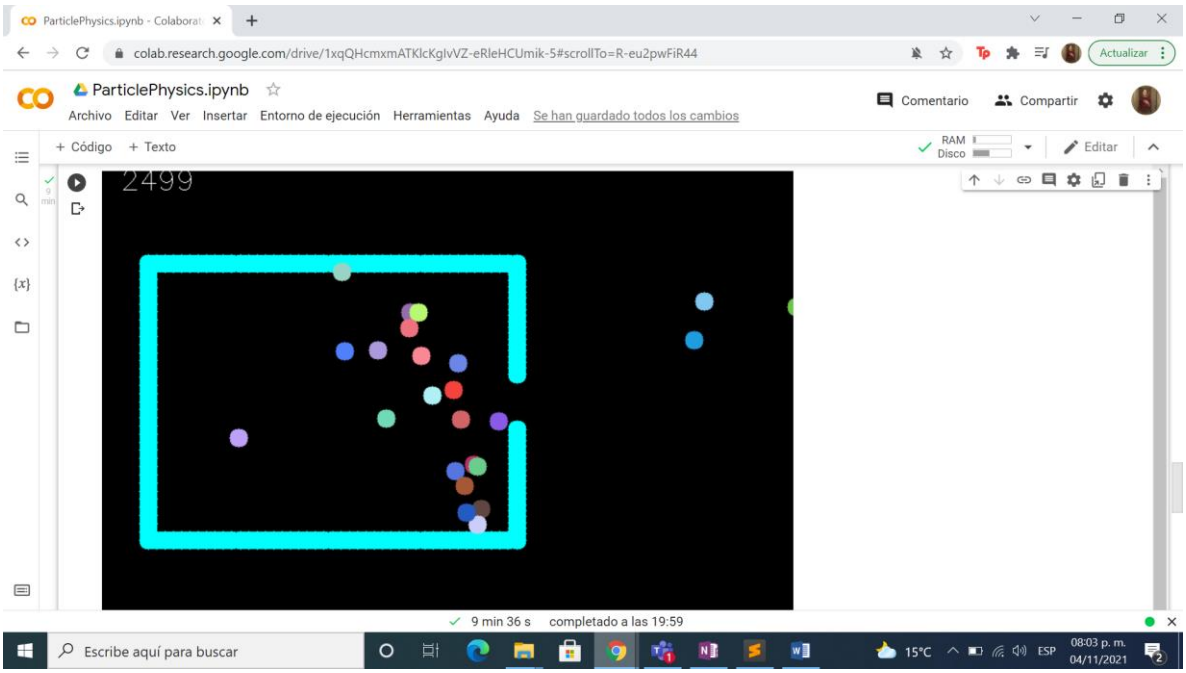


**Figura 17.** Captura de pantalla de un sistema de partículas con colisiones y una ventana de salida sin obstáculos en el camino al terminar 2500 iteraciones con 21 partículas y dos por salir.

El siguiente cambio fue implementar un obstáculo rumbo a la salida que separa un flujo grande de partículas que se dirigen hacia él, haciendo que el problema de que choquen constantemente unas con otras a tal grado de que se impiden el paso se considerablemente menor. Para aplicar este obstáculo construimos las paredes detalladas previamente en el desarrollo y se formó un rombo con estás.

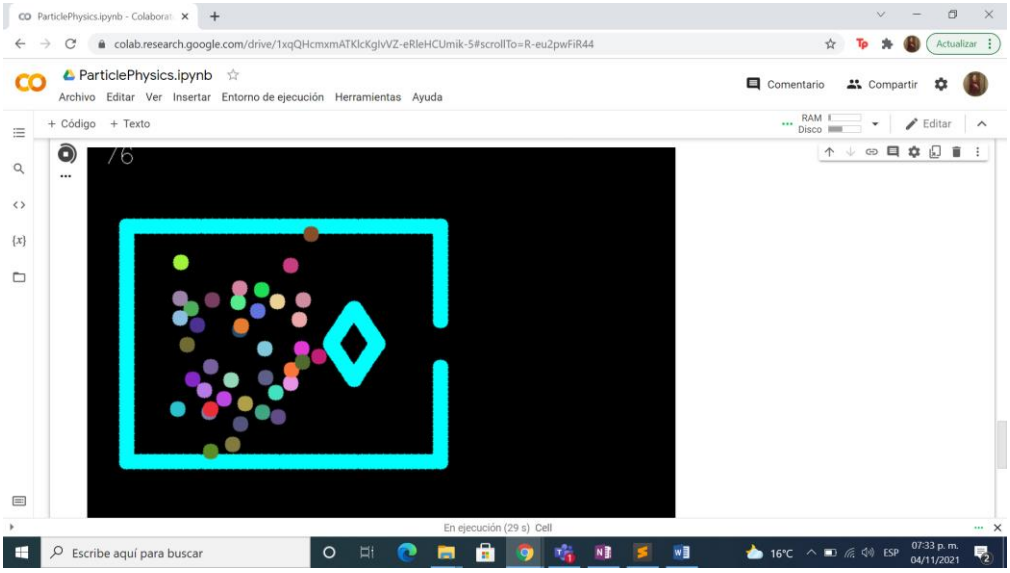


**Figura 18.** Captura de pantalla de un sistema de partículas con colisiones y una ventana de salida con un obstáculo en el camino en un inicio con 40 partículas dentro de las paredes.

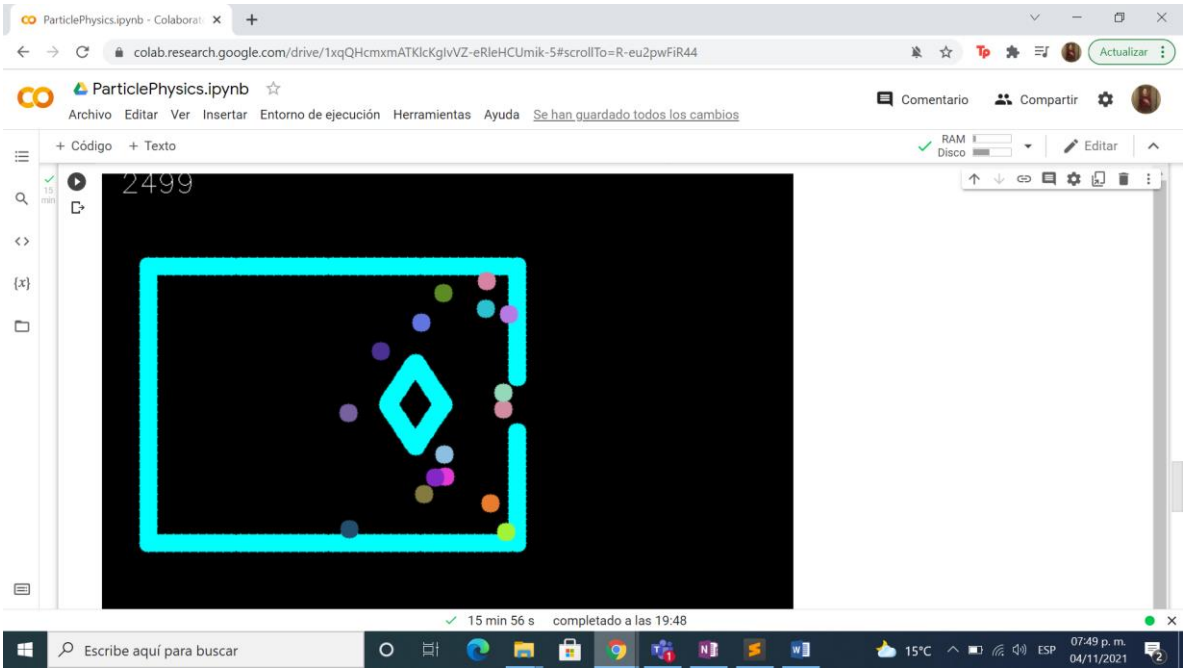


**Figura 19.** Captura de pantalla de un sistema de partículas con colisiones y una ventana de salida con un obstáculo en el camino al terminar 2500 iteraciones con 16 partículas y dos por salir.

Algo que pudo afectar que la inclusión del obstáculo en el sistema no mejorara de manera significativa la cantidad de partículas que logran escapar, puede ser la forma del obstáculo, ya que, si lo consideramos, cuando las partículas rebotan contra éste, no tendrán un vector que las dirija hacia esta, puede que la implementación de otras formas 9 mayor cantidad de obstáculos, pudieran usarse para diseñar un sistema que efectivamente tiene un mayor índice de efectividad.

## Conclusiones y recomendaciones

A lo largo de esta práctica pude trabajar bastante con sistemas de partículas y ver como pequeñas características en estos pueden cambiar enormemente la forma en que funcionan y lo que estos pueden representar. En la investigación del marco teórico ví que ésta clase de sistemas de partículas pueden representar objetos difusos que observamos en la naturaleza como podría ser el fuego o una cascada, pero después de realizar la práctica puedo imaginarlos también como una manera más abstracta de representar lo que podría ser un grupo de personas o animales. Como por ejemplo la ola, donde podemos ver que las partículas son influenciadas por el comportamiento de sus vecinos, así como en las famosas olas en los estadios de fútbol. Sin embargo, ésta influencia en los grupos de personas también es algo que pasa en muchos otros comportamientos como puede ser el bostezar.

Otra comparativa que se asemeja con el segundo sistema de partículas en donde hay colisiones cuando las partículas buscan salir de un lugar y se encuentran con obstáculos en el camino, es en lugares como antros para evitar estampidas en las salidas o aglomeraciones en las que sea difícil salir en casos de emergencias.

En general fue una práctica bastante interesante que ayuda a adentrarse en el funcionamiento de los sistemas de partículas y lo diversos que pueden ser; además de que motivó mucho a realizar distintas pruebas y experimentos con los sistemad para ver distintos comportamientos y como ciertas variables podían influenciar la forma en la que interactuaban las partículas. Algunas de éstas pruebas en el primer sistema fueron el probar con distintos inicios de las olas o el que varias partículas iniciaran la ola al mismo tiempo y ver el cambio de comportamiento que iba teniendo.

Por otro lado, en el segundo sistema lo que se probó fueron distintas condiciones para hacer que las partículas tuvieran una tendencia a irse hacia la izquierda (donde se encontraba la salida) y distintas fórmulas que el obstáculo podía tener como figuras irregulares que le diera cierta ventaja a los rebotes para dirigirse hacia la salida.

## Bibliografía

Khan Academy. (s. f.). *Intro to particle systems (article)*. Recuperado 3 de noviembre de 2021, de https://www.khanacademy.org/computing/computer-

programming/programming-natural-simulations/programming-particle-systems/a/intro-to-particle-systems

Cesium. (2015, 16 julio). *Introduction to Particle Systems*. Recuperado 3 de noviembre de 2021, de https://cesium.com/learn/cesiumjs-learn/cesiumjs-particle-systems/