תכנות בשפת ++C, אביב 2021

תרגיל בית מספר 3

נושא: סימולציית שולחן הוקי אוויר
דדליין: יום שבת, 15/05/2021, 23:59
הגשה ביחידים

בהצלחה רבה!

תיאור התרגיל

בתרגיל זה תממשו עולם מונחה עצמים של שולחן הוקי אוויר, הכולל דיסקיות וקירות. העולם יתבסס על מודל פיסיקלי של התנגשויות אלסטיות של הדיסקיות עם דיסקיות אחרות ועם הקירות. מטרת התרגיל היא משולשת – ראשית, אפיון מונחה עצמים של בעיה נתונה, שנית, תרגול תכנותי של מחלקות, בנאים/מפרקים, מתודות וכן העמסת פונקציות/אופרטורים, ושלישית, מימוש של סימולציה תלוית-זמן.

שולחן הוקי אוויר: אפיון

השולחן מוגדר כאוסף של דיסקיות וקירות בדו-מימד. לכל דיסקית יש מיקום, רדיוס מהירות ומסה, ולכל קיר יש שתי נקודות קצה אותם הוא מחבר בקו ישר. כל הדיסקיות מתקדמות בכוון ובמהירות קבועה עד להתנגשות של דיסקית עם קיר או עם דיסקית אחרת. במקרה של התנגשות עם דיסקית אחרת, שתי הדיסקיות משנות את מהירותן לפי הכללים של התנגשות אלסטית כמתואר בהמשך. אם דיסקית מתנגש בקיר, רק הדיסקית משנה את מהירותה, ומצב הקיר נותר ללא שינוי. הרעיון הכללי הינו לבנות את השולחן עם הדיסקיות והקירות, להריץ סימולצייה שלו במשך מספר יחידות זמן שייקבע, ולהדפיס כל יחידת זמן את מיקום כל הדיסקיות.

מצב השולחן וקידום הסימולציה בזמן

שולחן כולל מספר כלשהו של דיסקיות היכולות לזוז, ושל קירות קבועים.

הפרמטרים הנשמרים לכל דיסקית הם מיקום (x,y) ביחידות של סנטימטר, מהירות (vx,vy) ביחידות של סנטימטר לשנייה, רדיוס r בסנטימטרים, ומסה m בגרמים הנקבעת הנקבעת לפי הנוסחה $m = c\,r^2$. הקבוע c הנקבע על ידי צפיפות ועובי הדיסקיות, אינו ידוע, אבל זהה לכל הדיסקיות. אפשר לראות מהנוסחאות שערכו של c אינו משפיע על התנהגות הדיסקיות.

הפרמטרים הנשמרים לכל קיר הם שתי נקודות הקצה שלו (x1,y1) ו-(x2,y2).

יש לקרוא את מצבו ההתחלתי של השולחן מהקלט, וכן את זמן הסימולציה הנדרש T בשניות ואת יחידת הזמן לעדכון הסימולציה Δt.

על מנת לקדם את הסימולצייה בזמן Δt, יש לעבור על הדיסקיות אחת אחת לפי הסדר בו הוגדרו בקובץ הקלט, ועבור כל דיסקית d לבצע את הפעולות הבאות. (הפעולות המסומנות בקו תחתון יתוארו בהמשך):

1. לעדכן את מיקום הדיסקית – x += vx*Δt, y += vy*Δt
2. לבצע לולאה על כל הדיסקיות האחרות d'. <u>אם יש התנגשות בין d ו-d'</u> אז <u>יש לעדכן את מהירות שתי הדיסקיות</u>.
3. לבצע לולאה על כל הקירות w. <u>אם יש התנגשות בין d ו-w</u> אז <u>יש לעדכן את מהירות הדיסקית d</u>.
4. אם בשלבים 2 או 3 התגלתה התנגשות של d בדיסקית או קיר, אז יש להחזיר אותה למיקומה שלפני ביצוע שלב 1.

בדיקת התנגשות ועדכון מהירויות לאחר התנגשות

ההגדרות של פעולות על וקטורים דו-מימדיים מופיעות בנספח.

בדיקה אם דיסקית d שמרכזה p=(x,y) ורדיוסה r מתנגש בדיסקית d' שמרכזה p'=(x,y) ורדיוסה r':

$$\|p - p'\| \le r + r'$$

מציאת הנקודה הכי קרובה של הקטע p1=(x1,y1), p2=(x2,y2) לנקודה p=(x,y)

```
If p1 = p2    // segment has length zero
  return p1
else {
  α = (p − p1) · (p2 − p1)/‖p1 − p2‖²    // · is the dot product operator
  if α < 0
    return p1
else if α > 1
    return p2
else
    return p1 + α ∗ (p2 − p1)
```

בדיקה אם דיסקית d שמרכזה p=(x,y) ורדיוסה r מתנגשת בקיר w אשר קצותיו p1=(x1,y1), p2=(x2,y2).

```
Find the point q on the segment that is closest to p.
Check if ‖p − q‖ ≤ r.
```

עדכון מהירויות של שתי דיסקיות d,d' לאחר התנגשות. הדיסקית d עם מרכז p, מהירות v ומסה m, והדיסקית d' עם מרכז p', מהירות v' ומסה m'.

$$d = (p - p')/\|p - p'\| \quad \text{// normalized direction of collision}$$
$$\Delta = \big((v - v') \cdot d\big)\, d$$
$$v_{new} = v - \frac{2m'}{m + m'}\Delta$$
$$v'_{new} = v' + \frac{2m}{m + m'}\Delta$$

עדכון מהירות של דיסקית d לאחר התנגשות עם קיר. הדיסקית d עם מרכז p, מהירות v ומסה m, וקצות הקיר הם p1, p2.

```
Find the point q on the segment that is closest to p.
d = (p − q)/‖p − q‖    // normalized direction of collision
v_new = v − 2 (v · d) d
```

דרישות מימוש

אתם חופשיים לתכנן ולממש את התכנית כרצונכם, פרט לדרישות הבאות אותן עליכם לקיים במימוש שלכם:

- הדיסקיות צריכות להיות מופעים שונים (instances) של מחלקה Disc.
- הקירות צריכים להיות מופעים שונים (instances) של מחלקה Wall
- יש להגדיר מחלקה Vector2D המיצגת טיפוס של ווקטור בדו-מימד ולממש האופרטורים המתאימים על מנת לאפשר עבודה נוחה עם המחלקה במימוש תנועת הדיסקיות כפי שתוארה. במינימום, יש לממש על ידי העמסת אופרטורים +, -, *, +=, -=, =* את הפעולות של חיבור וחיסור וקטורים ושל כפל בסקלר.

פורמט הקלט

התכנית תקרא מ-stdin את הפקודות disc, wall, simulate. יכולות להיות מספר כלשהו של פקודות disc, wall אבל רק פקודת simulate אחת. לאחר simulate התוכנית תבצע את הסימולצייה היא תסיים את ריצתה בלי שתנסה לקרוא קלט נוסף. כל הארגומנטים של הפקודות הם מספרים מטיפוס double.

הפקודה disc מוסיפה דיסקה עם מיקום, מהירות ורדיוס נתונים לשולחן. פורמט הפקודה הינו:

```
disc <x> <y> <vx> <vy> <radius>
```

הפקודה wall מוסיפה קיר עם שני קצוות (x1,y1) ו-(x2,y2) לשולחן. פורמט הפקודה הינו:

```
wall <x1> <y1> <x2> <y2>
```

הפקודה simulate מבצעת את הסימולצייה והפרמטרים שלה הם משך הריצה של הסימולצייה בשניות T, ויחידת הזמן dT לעדכון הסימולצייה והדפסת מצב הסימולצייה.

simulate <T> <dT>

לאחר סיום הפקודה simulate יש לסיים את הסימולצייה ולא לנסות להמשיך לקרוא עוד פקודות.

פורמט הפלט

פלט הסימולצייה כולל שורה לכל נקודת זמן, מתחילת הסימולציה (זמן אפס) וכל פרק זמן dT עד סוף הסימולציה. פורמט השורה הינו כפי שנראה בדוגמא הבאה, כאשר <t> הינו זמן הסימולציה, והמשתנים <x1> ו-<y1> הם מיקום x,y של הדיסקה הראשונה, וכו'. יש לשים לב שמספר הזוגות x,y הינו לפי מספר הדיסקיות הכולל שהוגדר בקלט, וסדר ההדפסה מתאים לסדר שלהן בקובץ הקלט.

<t>: (<x1>,<y1>) (<x2>,<y2>)

ההדפסה של ערכי x,y הפלט הינה בפורמט fixed עם setprecision(4).

הודעות שגיאה

במקרה של שגיאה יש להדפיס את אחת מהודעות השגיאה הבאות ל-stderr ולסיים את הריצת התוכנית:

אם שם הפקודות לא חוקי, או שאי אפשר לקרוא את מספר הארגומנטים הנדרש, או שמגיעים לסוף הקובץ בלי פקודת simulate אזי יש להדפיס:

```
Error: illegal input.
```

בהוספת דיסקית/קיר יש לבדוק אם יש התנגשות עם עצם שכבר קיים על הלוח (דיסקית או קיר), ואם כן יש להוציא את אחת משתי הודעות השגיאה הבאות בהתאם למיקרה:

```
Error: disc to wall collision detected in initial configuration.
Error: disc to disc collision detected in initial configuration.
```

דוגמא

בהינן קובץ הקלט הבא:

```
disc 0 0 1 0.5 1
wall 5 -10 5 10
simulate 10 0.5
```

מוגדר לנו:

- דיסקה ברדיוס 1 אשר מרכזה ב-(0,0) ועם מהירות 1 בכוון החיובי של ציר ה-x.
- קיר הממוקם ב-x=5 מ-y=-10 עד y=10.
- הגדרת זמן סימולציה כולל של 10 שניות עם זמן עדכון כל חצי שנייה.

הפלט המתקבל הינו:

```
0: (0.0000,0.0000)
0.5000: (0.5000,0.2500)
1.0000: (1.0000,0.5000)
1.5000: (1.5000,0.7500)
2.0000: (2.0000,1.0000)
2.5000: (2.5000,1.2500)
3.0000: (3.0000,1.5000)
3.5000: (3.5000,1.7500)
4.0000: (3.5000,1.7500)
4.5000: (3.0000,2.0000)
5.0000: (2.5000,2.2500)
5.5000: (2.0000,2.5000)
6.0000: (1.5000,2.7500)
6.5000: (1.0000,3.0000)
7.0000: (0.5000,3.2500)
```

```
7.5000: (0.0000,3.5000)
8.0000: (-0.5000,3.7500)
8.5000: (-1.0000,4.0000)
9.0000: (-1.5000,4.2500)
9.5000: (-2.0000,4.5000)
10.0000: (-2.5000,4.7500)
```

אנו רואים בפלט את הדיסקה זזה מ-(0,0) בכוון החיובי של ציר ה-x, וכל חצי שנייה היא מתקדמת בחצי סנטימטר עד לזמן 3.5. בזמן זה, העדכון מקדם אותה למיקום (4,0) אשר נוגע בקיר. לפיכך מתבצע עדכון למהירות והיא הופכת להיות (-1,0) ומיקום הדיסקה נשאר ללא שינוי. לכן בזמן 4.0 המיקום נשאר זהה לזה שבזמן 3.5. מנקודה זו והלאה, כל חצי שנייה אנו רואים שהדיסקה זזה שמאלה בחצי סנטימטר עד סוף הסימולצייה.

נספח – פעולות וקטוריות בדו מימד

- חיבור: $(x1, y1) + (x2, y2) = (x1 + x2, y1 + y2)$.
- חיסור: $(x1, y1) - (x2, y2) = (x1 - x2, y1 - y2)$.
- כפל בקבוע: $c * (x, y) = (c * x, c * y)$.
- נורמה: $\|(x, y)\| = \sqrt{x^2 + y^2}$.
- מכפלת פנימית (dot product): $(x1, y1) \cdot (x2, y2) = x1 * x2 + y1 * y2$.

דגשים

- יש לתכנן מראש את מבנה התכנית, ולהגדיר בהתאם את האובייקטים איתם תעבדו; הקפידו על תכנון מונחה-עצמים של הסימולציה על מרכיביה השונים.
- הקדישו מחשבה למבני הנתונים השונים בהם תעשו שימוש; הביאו בחשבון שיקולי יעילות.
- עבור כל אובייקט יש לבחון מפורשות את הצורך במימוש "שלושת הגדולים" (*Rule of Three*), ולהתייחס בתיעוד למקרים בהם ברירת המחדל מספיקה.
- יש לבדוק תקינות קלטים ולהציג הודעות שגיאה מתאימות. **יתכן ובגלל העבודה עם double תהיה בעייה של אי-הסכמה קטנה עם קבצי הרפרנס היותר ארוכים**. במקרה כזה, אפשר לבדוק מהי אי ההסכמה באמצעות script פייתון שמצורף או בכל אמצעי אחר שנראה מתאים. הבודקים יודעים על הבעייה, ויערכו בהתאם.
- **בתרגיל בית זה אין להשתמש בספריית STL; מבני הנתונים והאלגוריתמים צריכים להיות ממומשים על-ידיכם**.
- עליכם לוודא כי התכנית עוברת קומפילציית g++ התואמת את הקומפיילר שעל שרת המכללה <u>ללא כל שגיאות או אזהרות כלשהן</u>, ורצה בהצלחה.
- עליכם לתעד את הקוד באמצעות הערות קצרות, אך משמעותיות, המתארות את הפונקציות השונות.
- בייחוד יש להקפיד על מימוש מחלקה של וקטורים דו-מימדיים עם כך האופרטורים הנלווים שמאפשרים עבודה נוחה עימה.
- תרגיל בית 4 (העוקב) יהווה המשך של תרגיל בית זה.
- <u>יש להריץ את הבודק האוטומטי על שרת החוג בטרם ההגשה בכדי לוודא תאימות ונכונות של ההגשה</u>: התחברו לשרת החוג והריצו **hwcheck** על הארכיב שלכם, או לחילופין העלו את הארכיב תוך שימוש בפרוטוקול HTML בקישור https://cs.telhai.ac.il/homework. לחילופין, התחברו לשרת ה-linux של המכללה, cs.telhai.ac.il והריצו את הפקודה hwcheck עם קובץ ה-zip שלכם כארגומנט.

הגשה

- עליכם להגיש במערכת Moodle קובץ ארכיב מטיפוס zip בלבד, ששמו כולל את קוד הקורס ('32'), שם התרגיל ('ex3') ותעודת הזהות של הסטודנט/ית המגיש/ה, מופרדים בקו תחתי בפורמט הבא: **32_ex3_studID.zip**
- על ארכיב zip זה להכיל את כל קבצי המקור (ממשק/מימוש) הנדרשים לקומפילציה, והוא רשאי להכיל תיעוד טקסטואלי; מבחינת טיפוסי קבצים, עליו לכלול רק קבצים עם סיומות ***.cpp *.h *.txt**
- לדוגמא: על סטודנט/ית בעל/ת מספר זיהוי 012345678 להגיש ארכיב בשם 32_ex3_012345678.zip הכולל את כל קבצי המקור של הפרוייקט, *ללא תיקיות כלשהן*, ורשאי להכיל קובץ טקסטואלי לתיעוד.

תכנות בשפת C++, אביב 2021

אי-הקפדה על ההנחיות, כולל פורמט ההגשה הדיגיטלי, תגרור הורדה בציון התרגיל.

לא תתקבלנה הגשות באיחור!