Разработайте программу, которая нарисует на холсте изображение Луны с заданной фазой. Фаза будет задаваться числом: 0 – новолуние, 0..1 – растущая Луна, 1 – полнолуние, 1..2 – убывающая Луна, 2 – опять новолуние (таблица 1).

Таблица 1. Как пронумерованы фазы Луны

0	0.5	1	1.5	2
	☽	○	☾	

1. Создайте окно с холстом размера 100x100.
2. Создайте глобальные переменные R = 40 для радиуса, и CENTER_X = 50, CENTER_Y = 50 для центра «Луны».
3. Создайте обработчик рисования, в котором нарисуйте круг с этими параметрами.
4. Создайте глобальную переменную для фазы phase.
5. Создайте поле ввода для фазы, с обработчиком, который записывает введённое значение в переменную phase (не забудьте преобразовать текст в float).
6. Теперь можно начать рисовать фазу в обработчике рисования. Сначала реализуйте рисование для растущей Луны.

   Чтобы получить неполную фазу надо закрасить часть уже нарисованной Луны чёрным. Лучше всего было бы применить для этого эллипс, но, к сожалению, в библиотеке Simplegui нет функции для рисования эллипсов. Поэтому придётся использовать многогранник в качестве приближения к эллипсу. Начните с очень грубого приближения – пятиугольника (рис. 1).

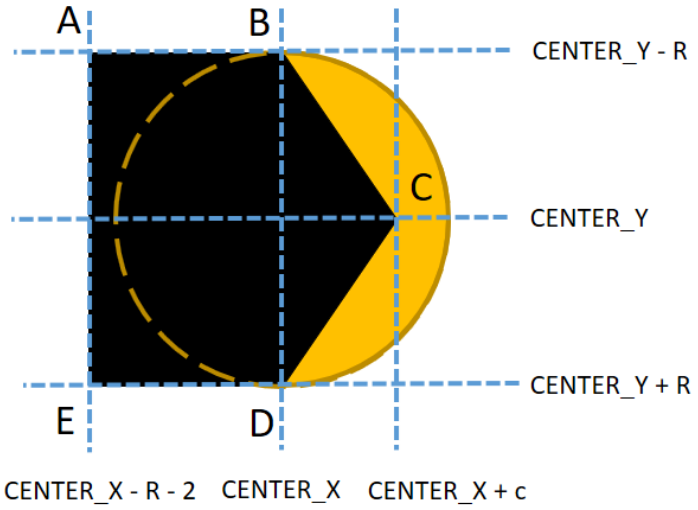

Рис. 1. Первое приближение к рисованию фазы луны

Для рисования пятиугольника воспользуйтесь функцией canvas.draw_polygon следующим образом[1]:

```
canvas.draw_polygon([[A_x, A_y], [B_x, B_y], [C_x, C_y],
                     [D_x, D_y], [E_x, E_y]], 1, 'black', 'black')
```

Точки B и D лежат на полюсах окружности, точки A и E должны находится немного левее её левого края, координаты этих точек задать не сложно.

[1] В языке Python можно разбить код на несколько строк, если в первой строке есть открывающаяся скобка, которая закроется в последней строке (любая: круглая или квадратная).

Положение точки C зависит от фазы, для расчёта её положения воспользуйтесь соотношением[2]

$$c = R * \cos(phase * \pi).$$

Чтобы использовать математические функции импортируйте библиотеку math, для чего необходимо написать «import math» в начале программы, после чего математические функции и константы доступны, например, как «math.cos(x)» и «math.pi». Обратите внимание, что тригонометрические функции традиционно работают в радианах.

7. Проверьте, что программа корректно рисует фазы растущей Луны для чисел в диапазоне от 0 до 1, в частности при фазе 0.5 должна быть нарисована ровная половина Луны.
8. Теперь реализуйте рисование для второй половины лунного цикла.
   При переходе ко второй половине цикла «тень» должна начать рисоваться справа, а точка C – вернуться на крайнее правое положение. Чтобы не дублировать одинаковый код для рисования в условном операторе, можно сделать следующее.
   Сначала вычислите значение
   $$s = \begin{cases} 1, & \text{если } \sin(phase * \pi) \geq 0, \\ -1 & \text{иначе.} \end{cases}$$
   Теперь достаточно вычислять абсциссы точек A и E как $CENTER\_X - s * (R + 2)$, а точки C как $CENTER\_X + c * s$.
9. Проверьте, что во второй половине цикла всё рисуется корректно.
10. Пора сделать линию на Луне более похожей на эллипс.
    Для этого реализуйте функцию shadow_point(y), которая вычисляла бы координаты точки F на границе скрываемой области (рис. 2).

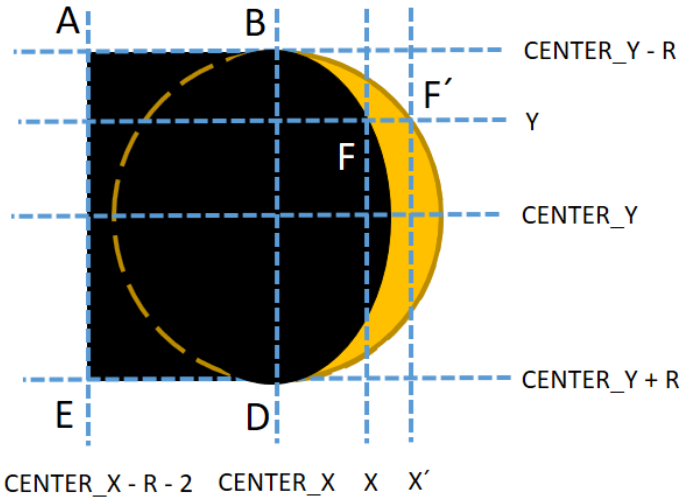

Рис. 2. Вычисление точки на границе

Функция будет получать на вход ординату Y для которой нужно рассчитать положение на границе закрашиваемой области. Для расчётов используйте глобальные переменные CENTER_X, CENTER_Y, phase и знания тригонометрии или теоремы Пифагора. Сначала рассчитайте ординату X’ пересечения прямой, проведённой на высоте Y, с окружностью Луны (точка F’ на рис. 2). Абсциссу X точки F можно рассчитать из соотношения $X - CENTER\_X = s * \cos(phase * \pi) * (X' - CENTER\_X)$.

Чтобы сделать код рисования более компактным, сделайте чтобы функция shadow_point()

[2] Освещённая Солнцем часть Луны — это всегда полусфера, но мы её видим с разных сторон и фактически требуется нарисовать проекцию этой полусферы на плоскость. Поэтому разумно в первом приближении допустить, что граница освещённой части движется по закону косинуса, хотя может оказаться что в настоящей астрономии всё немного сложнее.

возвращала сразу две координаты, для этого достаточно в конце функции написать

```
return [x, y]
```

В этом случае в коде рисования можно заменить взятые в квадратные скобки координаты точек на вызов функции:

```
canvas.draw_polygon([[A_x, A_y], [B_x, B_y],
                     shadow_point(CENTER_Y – R/2),
                     shadow_point(CENTER_Y),
                     shadow_point(CENTER_Y + R/2),
                     [D_x, D_y], [E_x, E_y]], 1, 'black', 'black')
```

В этом примере добавлено две промежуточных точки. Добавьте столько точек, чтобы картинка выглядела приемлемо и проверьте отображение на разных фазах.

11. На промежуточных фазах картинка должна выглядеть лучше, но «ровные» фазы новолуния (значения 0 и 2) и полнолуния (значение 1) пока содержат артефакты. Чтобы избавиться от них следует для новолуния не рисовать вообще ничего (включая исходную окружность), а для полнолуния – рисовать только окружность и ничего более.

    Чтобы достичь этого результата добавьте условия, используя ограничение на то, что модуль $\sin(phase * \pi)$ близок к нулю (точное значение порога определите экспериментально) – это обеспечит проверку на «ровную» фазу, а знак $\cos(phase * \pi)$ покажет, какая именно это фаза – новолуние или полнолуние.

    Для вычисления модуля используйте функцию math.abs().

12. Ну и наконец, можно добавить таймер, функция-обработчик которого будет увеличивать значение фазы на небольшое число, так чтобы получилась анимация.