

# Черепашача графіка у вебпереглядачі

Використання середовища програмування JavaScript Turtle Graphics Library

## Передмова

JavaScript Turtle Graphics Library - це середовище програмування, до складу якого входять:

- JavaScript-бібліотека TurtleGL.js, яка містить базовий набір інструментів для програмування Черепашачої 2D-графіки (Turtle Graphics) у вебпереглядачі, використовуючи об'єктно-орієнтований підхід;
- застосунок Інтерактивна мапа, який є зручним інструментом для створення прототипів зображень.

Мета створення середовища JavaScript Turtle Graphics Library - навчання основам програмування на прикладі роботи з Черепашачою 2D-графікою у вебпереглядачі, використовуючи мову програмування JavaScript і можливості бібліотеки [p5.js](#).

Версія посібника у формі інтерактивного покрокового підручника опублікована [ТУТ](#).

## Підключення бібліотеки TurtleGL.js

Бібліотека TurtleGL.js використовує засоби бібліотеки p5.js. Щоб підключити бібліотеку TurtleGL.js у свій проект, оберіть один із запропонованих способів:

1. В онлайн-редакторі [editor.p5js.org](#) файл бібліотеки TurtleGL.js необхідно підключити у файлі index.html перед файлом ескізу sketch.js.
2. У разі локальної розробки [завантажити](#) зразок проекту p5.js, у який підключити файл бібліотеки TurtleGL.js у файлі index.html перед файлом ескізу sketch.js.
3. У онлайн-редакторі [CodePen](#): -- перейти у вкладку JS; -- натиснути на зображення шестерні; -- у розділі Add External Scripts/Pens у рядку пошуку знайти і обрати p5.js; -- у розділі Add External Scripts/Pens натиснути на кнопку +add another resource і у рядку, що з'явився, додати посилання на файл бібліотеки TurtleGL.js; -- зберегти зміни, натиснувши кнопку Save & Close (для зареєстрованих користувачів), інакше - кнопку Close.
4. В онлайн-середовищі [Replit](#) при створенні нового Repl необхідно обрати шаблон p5.js, відвантажити файл бібліотеки TurtleGL.js у створений Repl і підключити файл бібліотеки у файлі index.html перед файлом ескізу script.js.
5. В онлайн-середовищі [OpenProcessing](#) у створений ескіз за допомогою розділу FILES відвантажити файл бібліотеки TurtleGL.js. Перейшовши у розділ SKETCH і обравши режим (MODE) HTML/CSS/JS, підключити відвантажений файл бібліотеки у файлі index.html перед файлом ескізу mySketch.js.
6. Використати [p5js-widget.pp.ua](#) - простий онлайн-редактор, створений на основі [p5.js-widget](#) - інструменту для вбудовування на сторінки сайтів редактора для запуску і редагування ескізів p5.js. У цей редактор бібліотека TurtleGL.js вже інтегрована і готова до використання.

Бібліотека TurtleGL.js реалізована у вигляді єдиного файлу, який можна завантажити у розділі [Початковий код бібліотеки TurtleGL.js](#).

Бібліотеку `TurtleGL.js` також можна використовувати у форматі [ескізу](#) (за потреби зробити `FORK` ескізу). У цьому разі жодних підключень бібліотеки виконувати не потрібно.

## Застосунок Інтерактивна мапа

Для зручності роботи з бібліотекою `TurtleGL.js` створений застосунок `Інтерактивна мапа`, за допомогою якого можна:

- відслідковувати, фіксувати на полотні і, за потреби, зберігати координати Черепашки у текстовий файл;
- зберігати у файл полотно з малюнком;
- змінювати властивості Черепашки/олівця (колір, форму, розмір, кутову орієнтацію Черепашки, товщину і колір олівця).

Застосунок `Інтерактивна мапа` опублікований:

- у форматі окремої [вебсторінки](#);
- у форматі [ескізу](#) (за потреби зробити `FORK` ескізу).

## Використання бібліотеки `TurtleGL.js`

### Початкові налаштування

У блоці `setup()` записуємо функцію `createCanvas(windowWidth, windowHeight)` для створення полотна, де `windowWidth` і `windowHeight` - це системні змінні, що містять значення ширина і висоти внутрішнього вікна (тобто вікна перегляду, у якому вебпереглядач «малює» вебсторінку) відповідно, і викликаємо `myCode()` - це назва блоку, в якому записані інструкції для Черепашки.

```
let t;  
  
function setup() {  
  createCanvas(windowWidth, windowHeight);  
  myCode();  
}
```

Значення розмірів полотна, які, за замовчуванням, використовує функція `createCanvas()` (коли викликається без аргументів) - `200x200` пікселів. За потреби можна створити полотно будь-якого розміру, зазначивши у `createCanvas()` відповідні значення розмірів ширини і висоти.

Також на початку у коді оголошуємо ім'я для майбутньої Черепашки - `t`.

Ім'я для Черепашки можна обрати на свій задум.

Блок `myCode()` має такий вигляд:

```
function myCode(){  
  // інструкції для Черепашки  
}
```

Усі інструкції для керування Черепашкою записуємо у блоці `myCode()`. Назву цього блоку можна змінити на свій задум.

Завжди першою інструкцією у блоці `myCode()` є інструкція зі створення об'єкта Черепашки з певним ім'ям (наприклад, `t`), оголошеним раніше:

```
function myCode(){
  t = new Turtle();
}
```

Отже, Черепашка отримала своє ім'я `t`. Тепер усі інструкції для Черепашки необхідно записувати у вигляді `t.інструкція`.

## Координатна сітка

Якщо необхідно створити координатну сітку на полотні, у блоці `myCode()` до Черепашки з ім'ям `t` застосовуємо інструкцію `grid()`:

```
function myCode(){
  t = new Turtle();
  t.grid();
}
```

Інструкція `grid()` використовує три параметри із такими значеннями за замовчуванням:

- крок сітки - 50 пікселів,
- колір сітки - [Platinum](#),
- колір осей - [Cerulean](#).

Значення кольору записується як рядок в одному із форматів: `"red"` (назва), `"#fdfd90"` (шістнадцяткове значення), `"rgb(11, 156, 78)"` (значення червоної, зеленої, синьої складових), `"rgba(45, 145, 67, 0.5)"` (значення червоної, зеленої, синьої складових і прозорості).

Значення за замовчуванням використовуються тоді, коли `grid()` викликається без аргументів, як у прикладі вище. За потреби `grid()` можна викликати із користувацькими значеннями, зазначивши їх у дужках у вказаному порядку. Наприклад:

```
function myCode(){
  t = new Turtle();
  t.grid(30, "rgba(43, 41, 70, 0.5)", "#ff9800"); // Space cadet, Orange peel
}
```

Використовуючи інструкцію `setStepGrid(step)` перед малюванням сітки, можна окремо встановити крок сітки `step`, а за допомогою інструкції `getStepGrid()` можна отримати поточне значення кроку сітки.

## Відображення Черепашки, інформаційна панель та компас

За замовчуванням, на полотні відображається інформаційна панель з даними про:

- стан Черепашки (кутову орієнтацію, поточні координати),
- стан олівця (на полотні чи піднятий),
- координати вказівника миші.

А у правому нижньому куті полотна увімкнений компас, який вказує напрямок і значення кутової орієнтації Черепашки.

За відображення вищезгаданих елементів інтерфейсу відповідають інструкції, які записуються у блоці `draw()`

```
function draw() {  
  t.place();  
  t.compass();  
  t.dashboard();  
}
```

і використовуються для таких цілей:

- `place()` - показати Черепашку на полотні у її поточних координатах. Цю інструкцію рекомендується завжди використовувати.
- `compass()` - показати компас.
- `dashboard()` - показати інформаційну панель.

Також на екрані можна відобразити ім'я розробника: у блоці `draw()` розмістити інструкцію `t.creator()`.

## Черепашка і p5.js

На полотні поруч з Черепашкою можна створювати зображення, використовуючи інструменти бібліотеки [p5.js](#).

Виклики функцій `p5.js` для цих цілей записуються усередині функцій `draw()` і `setup()`. Зверніть увагу, що, за замовчуванням, початок координат `(0, 0)` у бібліотеці `p5.js` розташований у лівому верхньому куті полотна.

Початок координат `(0, 0)` для Черепашки міститься в центрі полотна. За таких умов, одночасно відслідковувати координати для Черепашки і для побудови фігур складно.

У цьому разі інструкції для Черепашки і код для побудови фігур за допомогою інструментів бібліотеки `p5.js` можна записувати у блоці `myDraw()` (за потреби назву блоку можна змінити на іншу) між коментарями `// початок коду для фігур` і `// кінець коду для фігур`, а виклик `myDraw()` помістити у блоці `draw()`:

```
function draw() {  
  myDraw();  
  t.place();  
  t.compass();  
  t.dashboard();  
}  
  
function myDraw(){  
  push();  
  translate(width / 2, height / 2);  
  scale(1, -1);  
  
  // початок коду для фігур  
  let [x, y] = t.getPosition();  
  stroke(208, 85, 163); // mulberry  
  fill(255, 0);  
  circle(int(x), int(y), 100);  
  // кінець коду для фігур
```

```
pop();  
}
```

За допомогою виразу `let [x, y] = t.getPosition();` можна отримати поточні координати Черепашки з ім'ям `t` в координатній сітці, початок координат якої міститься в центрі полотна, і, за потреби, використати для побудови зображень фігур та створення анімаційних ефектів за допомогою інструментів бібліотеки `p5.js`.

У разі використання функції `background()` із бібліотеки `p5.js`, варто правильно зазначити місце її виклику у коді, щоб уникнути небажаного зафарбовування всього полотна. Таким місцем розташування виклику функції `background()` може бути тіло функції `myDraw()`.

Між коментарями `// початок коду для сітки і тла полотна` і `// кінець коду для сітки і тла полотна` також можна розмістити виклик інструкції для малювання сітки на полотні, коли необхідно одночасно використовувати і кольорове полотно, і координатну сітку.

```
function myDraw(){  
  // початок коду для сітки і тла полотна  
  background(70, 77, 119); // YInMn Blue  
  t.grid(30, "rgba(43, 41, 70, 0.5)", "#ff9800"); // Space cadet, Orange peel  
  // кінець коду для сітки і тла полотна  
  
  push();  
  translate(width / 2, height / 2);  
  scale(1, -1);  
  
  // початок коду для фігур  
  let [x, y] = t.getPosition();  
  stroke(208, 85, 163); // Mulberry  
  fill(255, 0);  
  circle(int(x), int(y), 100);  
  // кінець коду для фігур  
  
  pop();  
}
```

Завдяки тому, що виклик функції `background()` у тілі функції `myDraw()` розміщений найпершим, зафарбовування полотна не буде впливати на результати викликів інших функцій для малювання фігур.

## Багато Черепашок

Для створення двох (або більше) Черепашок, необхідно оголосити їхні імена та створити їх із цими іменами:

```
let t1, t2;  
  
function setup() {  
  createCanvas(windowWidth, windowHeight);  
  myCode();  
}  
  
function myCode(){
```

```
t1 = new Turtle();
t2 = new Turtle();
}
```

Відображення на полотні створених Черепашок відбувається за допомогою виклику інструкцій `place()` для кожної із них у блоці `draw()` :

```
function draw() {
  t1.place();
  t2.place();
}
```

Відповідно інструкції для різних Черепашок записуються у форматі `t1.інструкція` , `t2.інструкція` і т. д.

*У режимі кількох Черепашок інформаційна панель і компас показують дані для одного екземпляра Черепашки, для якого вони викликаються.*

### Зміна розмірів вікна полотна/вебпереглядача

При зміні розмірів вікна полотна/вебпереглядача усі елементи інтерфейсу залишаються на своїх місцях і з'являються смуги прокручування. Щоб ці елементи налаштувалися відповідно до нових розмірів, необхідно оновити вебсторінку. За потреби, перед цим збережіть свій код.

Якщо ви працюєте з інтерактивною мапою, спочатку встановіть розміри вікна вебпереглядача, а потім оновіть вебсторінку, щоб елементи інтерфейсу налаштувалися відповідно до нових розмірів.

## Інструкції

### Рух

*Рух Черепашки відбувається **без анімації**, тобто виконуються усі записані інструкції і Черепашка відразу розташовується на полотні у точці з кінцевими координатами.*

Після запуску Черепашка з'являється на полотні:

- у точці з координатами `(0, 0)` (центр полотна),
- "дивиться" праворуч,
- має початкову кутову орієнтацію, що вимірюється у градусах, `0°` .

Інструкція	Опис
<code>forward(distance)</code>	Перемістити Черепашку вперед на відстань <code>distance</code> , у напрямку, куди "дивиться" Черепашка.
<code>back(distance)</code>	Перемістити Черепашку назад на відстань <code>distance</code> , у протилежний бік напрямку її руху.
<code>goto(x, y)</code>	Перемістити Черепашку в точку з координатами <code>(x, y)</code> .
<code>home()</code>	Перемістити Черепашку в точку з координатами <code>(0, 0)</code> (центр полотна) і встановити початкову кутову орієнтацію.
<code>left(angle)</code>	Повернути Черепашку ліворуч на кут <code>angle</code> , де <code>angle</code> - число (ціле чи з плаваючою крапкою). Якщо значення кута <code>angle</code> додатне, то

	Черепашка повертається на це значення кута проти годинникової стрілки (кутова орієнтація стає <code>angle</code> ), якщо від'ємне - Черепашка повертається за годинниковою стрілкою на це значення кута (кутова орієнтація стає $360 - \text{abs}(\text{angle})$ , де $\text{abs}(\text{angle})$ - значення <code>angle</code> по модулю).
<code>right(angle)</code>	Повернути Черепашку праворуч на кут <code>angle</code> , де <code>angle</code> - число (ціле чи з плаваючою крапкою). Якщо значення кута <code>angle</code> від'ємне, то Черепашка повертається на це значення кута проти годинникової стрілки (кутова орієнтація стає <code>angle</code> ), якщо додатне - Черепашка повертається за годинниковою стрілкою на це значення кута (кутова орієнтація стає $360 - \text{abs}(\text{angle})$ , де $\text{abs}(\text{angle})$ - значення <code>angle</code> по модулю).
<code>setHeading(angle)</code>	Встановити кутову орієнтацію Черепашки на кут <code>angle</code> , де <code>angle</code> - число (ціле чи з плаваючою крапкою). Якщо значення кута <code>angle</code> додатне, то Черепашка повертається на це значення кута проти годинникової стрілки (кутова орієнтація стає <code>angle</code> ), якщо від'ємне - Черепашка повертається за годинниковою стрілкою на це значення кута (кутова орієнтація стає $360 - \text{abs}(\text{angle})$ , де $\text{abs}(\text{angle})$ - значення <code>angle</code> по модулю).

## Черепашка

Інструкція	Опис
<code>getPosition()</code>	Отримати поточні координати Черепашки у вигляді масиву <code>[x, y]</code> .
<code>getHeading()</code>	Отримати поточну кутову орієнтацію Черепашки.
<code>setShape(shape)</code>	Встановити форму <code>shape</code> для Черепашки. <code>shape</code> може набувати значень: "blank" (невидимість), "circle", "square", "triangle". За замовчуванням, форма Черепашки "triangle".
<code>setShapeSize(s)</code>	Встановити розмір <code>s</code> для форми Черепашки. За замовчуванням, мінімальний розмір - 1, максимальне значення - 20.
<code>showTurtle()</code>	Зробити Черепашку видимою.
<code>hideTurtle()</code>	Зробити Черепашку невидимою.
<code>clr()</code>	Очистити полотно. Черепашка залишається у поточній точці, її кутова орієнтація зберігається.

## Олівець

Якщо олівець на полотні, при зміні координат Черепашка малюватиме лінію.

Інструкція	Опис
<code>penUp()</code>	Підняти олівець.
<code>penDown()</code>	Опустити олівець на полотно. За замовчуванням, олівець на полотні.
<code>setPenSize(s)</code>	Встановити у пікселях товщину лінії олівця на <code>s</code> . За замовчуванням, товщина лінії олівця 1 піксель.

### Колір

Значення кольору записується як рядок в одному із форматів: "red" (назва), "#fd9d90" (шістнадцяткове значення), "rgb(11, 156, 78)" (значення червоної, зеленої, синьої складових), "rgba(45, 145, 67, 0.5)" (значення червоної, зеленої, синьої складових і прозорості).

Інструкція	Опис
setColor(c, f)	Встановити колір c для олівця та колір заливки f для зафарбовування фігур. За замовчуванням, колір олівця - чорний, колір заливки - прозорий. Для встановлення лише кольору олівця інструкція використовується з одним параметром: setColor(c) .
setBgColor(c)	Встановити колір c для тла полотна. За замовчуванням, білий.
setFillColor(f)	Встановити колір заливки f для зафарбовування фігур. За замовчуванням, заливка прозора.
beginFill()	Увімкнути зафарбовування фігури поточним кольором заливки.
endFill()	Вимкнути зафарбовування фігури поточним кольором заливки.

### Фігури

Інструкція	Опис
oval(r, e)	Намалювати коло чи еліпс. Якщо параметри r і e набувають однакових значень, то Черепашка малює коло зі значенням радіуса, що дорівнює r. У разі різних значень r і e - отримуємо еліпс.
polygon([[x1, y1], [x2, y2], [x3, y3], ...])	Намалювати багатокутник за координатами вершин x1, y1, x2, y2, x3, y3, ... (за годинниковою стрілкою).

### Текст

Інструкція	Опис
write("txt", {horizontal: A, vertical: B}, {font: C, size: D, style: E})	Написати текст "txt" у поточній позиції Черепашки. Об'єкт {horizontal: A, vertical: B} використовується для вирівнювання тексту, де A може набувати значень LEFT, CENTER або RIGHT, а B - TOP, BOTTOM, CENTER або BASELINE. Об'єкт {font: C, size: D, style: E} визначає шрифт, розмір тексту і стиль відповідно, а саме: C може набувати значень Arial, Times, Verdana тощо, D - ціле число, E може набувати значень NORMAL, ITALIC, BOLD, BOLDITALIC.

За замовчуванням, текст є порожнім рядком і має такі параметри: вирівнювання {horizontal: CENTER, vertical: CENTER} і шрифту {font: "sans-serif", size: 12, style: NORMAL} .

### Розробка



Для розробки бібліотеки `TurtleGL.js` та застосунку `Інтерактивна мапа` використовувалась мова програмування `JavaScript` та застосовувались інструменти бібліотеки [p5.js](#).

## Початковий код бібліотеки `TurtleGL.js`

Початковий код бібліотеки `TurtleGL.js` зберігається в єдиному файлі, який можна завантажити [тут](#).

## Ліцензія

Використання бібліотеки `TurtleGL.js` визначається умовами ліцензії `GNU General Public License (GPL) version 3`.

*Код бібліотеки `TurtleGL.js` можна вільно і безкоштовно копіювати, розповсюджувати і змінювати на свій задум.*