



# Girls Who Code At Home

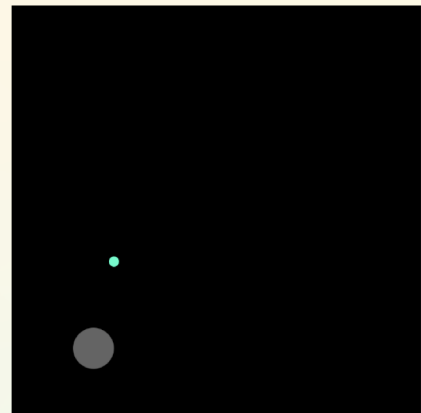
उल्का पकड़ो गेम: भाग 4

उल्का पकड़ें

## गतिविधि अवलोकन

भाग 3 में आपने जाना कि स्क्रीन पर उल्का को एक निर्धारित चाल से चलाने के लिए p5.js में चर राशि किस प्रकार बनाते और प्रयोग करते हैं। अब परस्पर-व्यवहार (इंटरैक्शन) जोड़ने का समय है! इस भाग में आप जानेंगे कि माउस के संचलन पर प्रतिक्रिया देने वाले एक कैचर को प्रोग्राम करके खिलाड़ी के इनपुट को कैसे शामिल करें। यह कैचर एक पाराभासी सफेद दीर्घवृत्त है जो माउस के साथ-साथ चलता है। गतिविधि के अंत तक पहुंचने पर आप जो कुछ सीखेंगे उसके पूर्वावलोकन के लिए [यहां](#) क्लिक करें।

यह गतिविधि आरंभ करने से पहले यह आवश्यक है कि आपने **उल्का पकड़ो गेम सीरीज़** का [भाग 1](#), [भाग 2](#), और [भाग 3](#) पूरा कर लिया हो।



## सीखने के लक्ष्य

इस गतिविधि को पूरा कर लेने पर आप निम्नलिखित कर सकेंगे...

- ❑ यह वर्णन करना कि किसी प्रोग्राम में किसी बुनियादी संचलन की नकल कैसे करें
- ❑ चर राशि और गणितीय संक्रियकों का उपयोग करके घटकों में विभिन्न व्यवहारों की प्रोग्रामिंग करना।

## सामग्रियां

- [p5.js ऑनलाइन एडिटर](#)
- [उल्का पकड़ो गेम सैंपल प्रोजेक्ट](#)
- [उल्का पकड़ो गेम भाग 4 संदर्भ मार्गदर्शिका](#)



## वुमन इन टेक स्पाटलाइट: रिबेका कोहेन पलासियोस (Rebecca Cohen Palacios)



तस्वीर स्रोत:  
[GamesIndustry.biz](http://GamesIndustry.biz)

यदि आपने इनमें से कोई वीडियो गेम खेला है – एल्डर स्कॉल्स: ब्लेड्स (Elder Scrolls: Blades), असेसिन्स क्रीड ओरिजिन्स (Assassin's Creed Origins), असेसिन्स क्रीड सिंडिकेट (Assassin's Creed Syndicate), या शेप अप (काइनेक्ट) (Shape Up (Kinect)) – तो आपने वह कुछ कार्य देखा ही होगा जिसका योगदान रिबेका ने इन गेम्स की रचना में दिया है! गेमिंग उद्योग में पहले से कहीं अधिक महिलाएं कार्य कर रही हैं, और इसका कुछ श्रेय रिबेका के कार्य को और उनके द्वारा तान्या शॉर्ट (Tanya Short) के साथ मिलकर स्थापित किए गए संगठन, [पिक्सेल्स \(Pixelles\)](#) के कार्य को जाता है।

पिक्सेल्स (Pixelles) एक अलाभ संगठन है जो गेम विकास के क्षेत्र में अधिकाधिक महिलाओं को प्रेरित करने के लिए समर्पित है। रिबेका ने पाया कि वीडियो गेम्स को अभी-भी “लड़कों वाली चीज़” के रूप में देखे जाने के साथ-साथ, गेमिंग उद्योग में ऐसी महिलाओं का एक विशाल प्रतिशत भी है जो पूर्वग्रह (बायस), सहयोग के अभाव, और महिलाओं, अश्वेतों और/या अल्पसंख्यकों के प्रति भेदभावपूर्ण व्यवहार के कारण इस क्षेत्र को अलविदा कह देता है। पिक्सेल्स (Pixelles) इन मुद्दों से निपटने के लिए आकांक्षी और करियर-मध्य में मौजूद महिलाओं को निःशुल्क मासिक कार्यशालाएं, मेंटरशिप, “अपना पहला गेम बनाएं” कार्यक्रम, करियर गतिवर्धक, सामाजिक नेटवर्किंग, एवं कई अन्य चीजें

प्रदान करता है... पिक्सेल्स (Pixelles) का सह-निर्देशन करने के साथ-साथ रिबेका वर्तमान में [बिहेवियर इंटरैक्टिव \(Behaviour Interactive\)](#) नामक कंपनी में सीनियर UI डिज़ाइनर के रूप में भी कार्य करती हैं। गेमिंग उद्योग में ऊबिसॉफ़्ट (Ubisoft) से अपनी शुरुआत करने से पहले रिबेका ने ग्राफिक और वेब डेवलपर के रूप में 6 वर्ष बिताए थे।

रिबेका के बारे में और, [पिक्सेल्स \(Pixelles\)](#) के साथ उनका जो कार्य है वह गेमिंग उद्योग में लैंगिक फासले को भरने के लिए किस प्रकार प्रयास करता है इस बारे में और जानें!

- ["पिक्सेल्स \(Pixelles\) करियर-मध्य में मौजूद माताओं को गेम्स में बने रहने में मदद कर रहा है"](#)
- ["महिलाओं द्वारा गेम विकास छोड़ने के शीर्ष 7 कारण"](#)
- ["मॉन्ट्रियल स्थित यह अलाभ संगठन महिलाओं को पुरुषों के वर्चस्व वाले वीडियो गेम उद्योग में घुसपैठ करने का मौका देता है"](#)
- ["गेम विकास के जरिए सशक्तीकरण": पिक्सेल्स गेम इन्क्यूबेटर"](#)

## झलक

एक कंप्यूटर वैज्ञानिक होना, कोडिंग में बेहतरीन होने की तुलना में अधिक है। इस बात के बारे में सोचने में थोड़ा समय बिताएं कि कैसे रेबेका और उनका काम उन शक्तियों से संबंधित है जिन पर महान कंप्यूटर वैज्ञानिक - बहादुरी, लचीलेपन, रचनात्मकता और उद्देश्य के निर्माण के दौरान ध्यान केंद्रित करते हैं।



प्रतिस्थितिव  
(रिज़िलिएंस)

गेमिंग उद्योग में महिलाओं को किन संघर्ष का सामना करना पड़ता है? पिक्सेल्स (Pixelles) महिलाओं को उनके करियर के संरक्षण में किस प्रकार सहायता देता है?

परिवार के किसी सदस्य या मित्र के साथ अपनी प्रतिक्रियाएँ साझा करें। चर्चा में शामिल होने हेतु दूसरों को रेबेका के बारे में अधिक पढ़ने के लिए प्रोत्साहित करें!

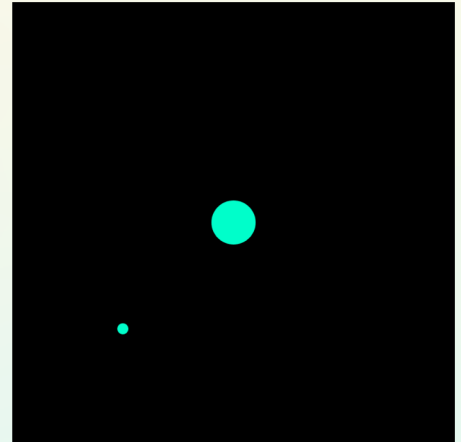
## चरण 1: कैचर जोड़ें (5-10 मिनट)

### कैचर बनाएं (3-5 मिनट)

सबसे पहले, हमें `ellipse()` फंक्शन का उपयोग करके स्क्रीन पर कैचर बनाना होगा। जैसा हमने उल्का के मामले में किया था, हम कैचर की चौड़ाई और ऊंचाई स्टोर करने के लिए चर राशि का उपयोग करेंगे। हम अगले चरण में x और y स्थान के लिए विशेष चर राशि का उपयोग करेंगे, ताकि हमें इन वैल्यूज़ को स्टोर करने के लिए चर राशि न बनाने पड़ें।

- ❑ हमारे कैचर की चौड़ाई और ऊंचाई स्टोर करने के लिए `setup()` के ऊपर (यानि उससे पहले) एक नया चर राशि जोड़ें। चूंकि हमारा कैचर एक गोला है, अतः हम दोनों के लिए समान कीमत प्रयोग करेंगे। हमने हमारे चर राशि को `catcherDiameter` नाम दिया है, पर आप इसे जो चाहें वो नाम दे सकते हैं। बस इतना याद रखें कि आपको अपने बाकी के पूरे कोड में वही नाम प्रयोग करना है।
- ❑ अपने कैचर के चौड़ाई व ऊंचाई चर राशि को 40 की कीमत निर्धारित करें।
- ❑ `draw()` फंक्शन के अंदर, `ellipse()` फंक्शन का उपयोग करके कैचर बनाएं। इसे उल्का वाले अपने कोड के नीचे जोड़ें। खुद को यह याद दिलाने के लिए एक छोटा सा कमेंट जोड़ें कि यह कैचर है।
- ❑ x और y पैरामीटर्स को 200 पर सेट करें, और फिर चौड़ाई और ऊंचाई के पैरामीटर्स को उस चर राशि पर सेट करें जो आपने ऊपर बनाया था।
- ❑ अपना कोड चलाएं।

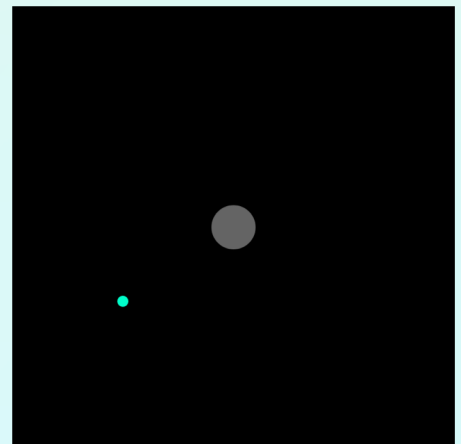
आपको कैनवास के बीच में एक नील-हरित गोला दिखना चाहिए:



### कैचर का रंग बदलें (3-5 मिनट)

हम हमारे कैचर को सफेद और अर्द्ध-पारदर्शी बनाना चाहते हैं ताकि हम हमारी उल्का को कैचर में जाते देख सकें। यानि हमें हमारे `fill()` फंक्शन का एक बार फिर प्रयोग करना होगा! पहले हमने बताया था कि `fill()` फंक्शन उस कमांड के नीचे आने वाली सभी आकृतियों में रंग भर देगा, बशर्ते कि आप प्रोग्राम को कोई और रंग भरने को न कहें - यह कुछ यूं है मानो आप पेंटब्रश को साफ करके उसे किसी और रंग में डुबो दें।

- ❑ कैचर दीर्घवृत्त के ऊपर एक नया `fill()` फंक्शन बनाएं। यह इससे नीचे की सभी आकृतियों पर नया रंग लागू कर देगा।
- ❑ RGB किमतें जोड़ कर अपने कैचर को सफेद बनाएं।
- ❑ पारदर्शिता को नियंत्रित करने के लिए एक चौथा पैरामीटर जोड़ें और उसकी कीमत 100 पर सेट कर दें। इस वैकल्पिक पैरामीटर को एल्फा कीमत कहते हैं। आप 0 से 255 के बीच की कोई भी कीमत सेट कर सकते हैं जहां 0 का अर्थ पूरी तरह पारदर्शी है और 255 का अर्थ पूरी तरह अपारदर्शी। यह कीमत कैचर के प्रदर्शन को किस प्रकार बदलती है यह समझने के लिए आप कुछ अलग-अलग कीमतों के साथ आजमाइश कर सकते हैं।
- ❑ अपना कोड चलाएं।



आपको कैचर का रंग बदलकर अर्द्ध-पारदर्शी सफेद होता दिखना चाहिए। नीचे वाले चित्र में कैचर ग्रे (धूसर) दिख सकता है। चूंकि यह अर्द्ध-पारदर्शी है, अतः बैकग्राउंड का थोड़ा सा रंग इसमें मिल जाता है।

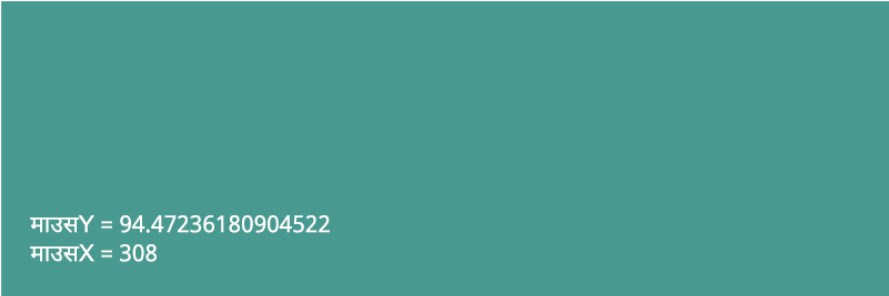




## चरण 2: खिलाड़ी का इनपुट जोड़ें (2-4 मिनट)

इस समय, कैचर कोई खास धर-पकड़ नहीं कर रहा है। हम चाहते हैं कि खिलाड़ी माउस का उपयोग करके इसके संचलन का नियंत्रण कर पाए। यानि हमें कैचर की x और y कीमत को इस प्रकार बदलने का एक तरीका ढूंढना होगा जिससे वे माउस की x और y कीमतों से मेल खाएं।

भाग्य से हमारे पास p5.js में ऐसे दो चर राशि पहले से हैं जो यही काम करते हैं! `mouseX` और `mouseY` नामक चर राशि में माउस की क्षैतिज और ऊर्ध्व स्थिति होती है। [उदाहरण रेखा-चित्र](#) में अपना माउस चला कर देखें कि ये कीमतें किस प्रकार बदलती हैं।



```
माउसY = 94.47236180904522  
माउसX = 308
```

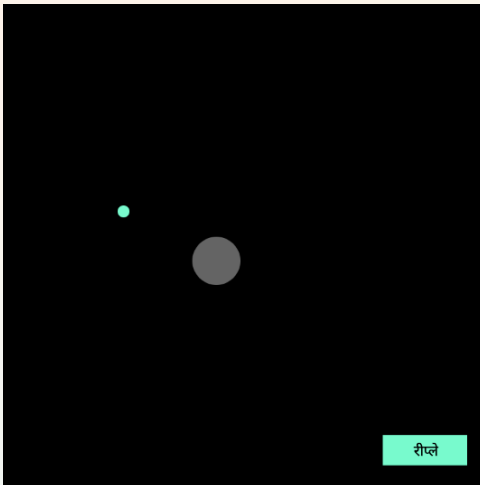
हम `mouseX` और `mouseY` का उपयोग करके कैचर को माउस से जोड़ सकते हैं। यानि माउस की x और y स्थितियां, कैचर की x और y स्थितियां बन जाती हैं। चलिए हमारे कैचर को अपडेट करते हैं:

- ❑ अपने कैचर दीर्घवृत्त की वर्तमान x कीमत के स्थान पर `mouseX` रखें।
- ❑ अपने कैचर दीर्घवृत्त की वर्तमान y कीमत के स्थान पर `mouseY` रखें।

## चरण 3: अपने कोड को परखें (2-5 मिनट)

चलिए अब तक हमने जो लिखा है उसे परख कर यह सुनिश्चित करते हैं कि हमारा प्रोग्राम उसी तरह चल रहा हो जैसे हम चाहते हैं। प्ले बटन पर क्लिक करके अपना रेखा-चित्र चलाएं। आपके सामने होना चाहिए:

- आपके माउस के संचलन के साथ-साथ चलने वाला कैचर।
- कैचर को सफेद रंग का और अर्द्ध-पारदर्शी होना चाहिए।
- आपके सामने रीप्ले बटन नहीं होना चाहिए।



उदाहरण रेखा-चित्र चलाने के लिए [यहां](#) क्लिक करें।

**ध्यान दें:** हमने एक रीप्ले बटन दिया है ताकि आप उल्का के व्यवहार को रीसेट कर सकें। यदि हमने यह नहीं दिया होता, तो आपको उल्का के स्क्रीन के निचले भाग से गिर जाने के बाद बस एक काला बॉक्स दिखता। हम अगले भाग में एक कंडीशनल की मदद से इसे ठीक कर देंगे, पर हम अभी वहां नहीं पहुंचे हैं!

**जैसा चाहा था वैसे कार्य नहीं कर रहा? ये डीबगिंग के सुझाव आजमाएं:**

- क्या आपका कोड सही मझाले कोष्ठकों (कलर्ली ब्रैकेट्स) के अंदर है?
- क्या आपने हरेक कोड की पंक्ति के अंत में सेमीकोलन टाइप किए हैं?
- क्या आपने चर राशियों और फंक्शन के नामों की सही स्पेलिंग लिखी? याद रखें कि जावास्क्रिप्ट केस सेंसिटिव है यानी इसमें अपरकेस और लोअरकेस अक्षरों को समान नहीं माना जाता है!
- क्या आपके फंक्शन्स सही स्थान पर और सही क्रम में हैं? याद रखें कि प्रोग्राम प्लो में क्रम मायने रखता है!
- क्या आपने अपनी उल्का और अपने कैचर, दोनों के लिए `ellipse()` फंक्शन के ऊपर अलग-अलग `fill()` फंक्शन्स लिखे हैं?
- यदि आपका कैचर दिखाई न दे, तो एल्फा कीमत को बढ़ाएं।
- क्या आपने `mouseX` और `mouseY` को अपने कैचर दीर्घवृत्त में सही पैरामीटर स्थान में जोड़ा था?



अपने कोड को पृष्ठ 3 पर संदर्भ मार्गदर्शिका से जांचना न भूलें।

## चरण 4: समझ की जाँच करें

वर्णन करें कि कैसे यह कोड की पंक्ति हमारे कैचर के व्यवहार को बदलेगी:

```
ellipse(200, 200, mouseX, mouseY);
```

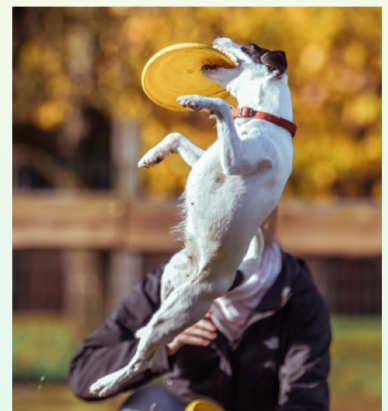


अपने कोड को पृष्ठ 3 पर संदर्भ मार्गदर्शिका से जांचना न भूलें।

## चरण 5: “पकड़ने” की संकल्पना करें (2 मिनट)

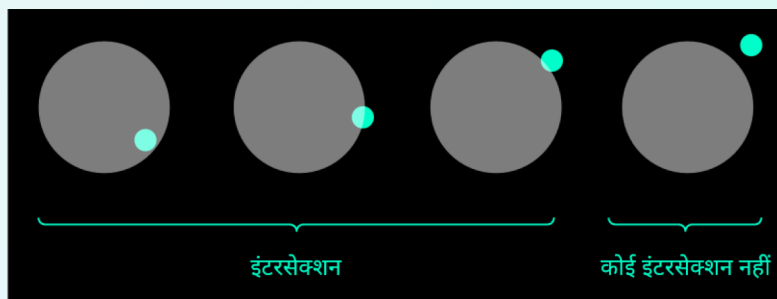
पिछले चरण में हमने खिलाड़ी को हमारे गेम के साथ परस्पर व्यवहार यानि इंटरैक्ट करवाने का एक तरीका खोज निकाला था। अब हम यह पता लगाना चाहते हैं कि गेम के विभिन्न घटकों को एक-दूसरे के साथ व्यवहार कैसे करवाएं। हम यह ज्ञात करना चाहते हैं कि कब कैचर ने उल्का को “पकड़ लिया” और कब उल्का स्क्रीन के निचले भाग से “जा टकराई”।

जब आप किसी चीज़ को पकड़ लेते हैं तो असल में क्या होता है? किसी गेंद को अपने दोस्त की ओर फेंकने या फ़र्श पर लुढ़काकर उसकी तरफ़ भेजने की कल्पना करें। जब वह उसके शरीर को छूती है और वह उसे कब्जे में ले लेता है, तो हम कहते हैं कि उसने गेंद पकड़ ली है। हम यह भी कह सकते हैं कि गेंद उसकी हथेली या पंजे (या चंगुल) से जा टकराई है या उसे छू गई है।



तस्वीर स्रोत: [Pixabay](#)

हम ऐसा कोड लिखना चाहते हैं जो परखेगा कि उल्का और कैचर ने एक-दूसरे को छुआ है या नहीं। इसे कोलिज़न डिटेक्शन (टक्कर का पता लगाना) भी कहते हैं, इस शब्द का अर्थ यह पता लगाने से है कि कब कोई दो आकृतियां एक-दूसरे को छूती हैं। हमारे प्रोग्राम में कोलिज़न डिटेक्शन के लिए हम इन कुछ तरीकों का उपयोग कर सकते हैं:



चलिए प्रोग्राम के अंदर हमारी पहुंच में मौजूद उस जानकारी के बारे में सोचते हैं जिससे हमें मदद मिल सकती है: हमें उल्का के x और y स्थान, कैचर के x और y स्थान, तथा दोनों का आकार ज्ञात है। हालांकि उनके स्थान लगातार बदलते रहते हैं, पर हमारे लिए वे कीमतें उनके x और y चर राशि के जरिए उपलब्ध रहती हैं। एक बार फिर, चर राशियां ही हमारे तारणहार हैं!

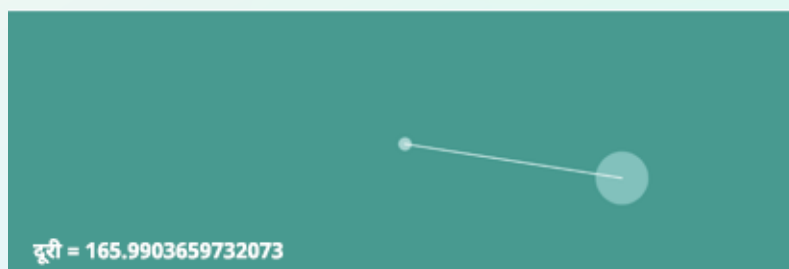
## चरण 6: दूरी की गणना करना (10-15 मिनट)

### `dist()` फंक्शन से मिलें (3-5 मिनट)

हम ऊपर वाली जानकारी का उपयोग करके यह गणना कर सकते हैं कि किसी क्षण विशेष पर कैचर, उल्का से कितनी दूर है। p5 में एक ऐसा फंक्शन है जो हमारे लिए ये गणनाएं करता है: `dist()` फंक्शन। यह फंक्शन दो बिंदुओं के बीच की दूरी की गणना करता है। हमें इसे बस पैरामीटर्स देने हैं! यह रही इसकी वाक्य-रचना:

जावास्क्रिप्ट	विवरण
<code>dist(x1, y1, x2, y2);</code>	<ul style="list-style-type: none"><li>→ <code>dist</code>: फंक्शन का नाम।</li><li>→ <code>()</code>: हम कोष्ठकों द्वारा हमारे प्रोग्राम को यह बताते हैं कि उसे इस फंक्शन को कॉल करना है। कभी-कभी हम हमारे कोष्ठकों के अंदर फंक्शन के पैरामीटर या इनपुट शामिल कर देते हैं।</li><li>→ <code>x1</code>: पहले बिंदु का x कोऑर्डिनेट।</li><li>→ <code>y1</code>: पहले बिंदु का y कोऑर्डिनेट।</li><li>→ <code>x2</code>: दूसरे बिंदु का x कोऑर्डिनेट।</li><li>→ <code>y2</code>: दूसरे बिंदु का y कोऑर्डिनेट।</li><li>→ <code>::</code> p5.js की सभी कोड की पंक्तियों का अंत सेमीकोलन से होना आवश्यक होता है।</li></ul>

**डेमो रेखा-चित्र** देखें। टेक्स्ट वह कीमत दर्शाता है जो `dist()` फंक्शन ने बीच वाले गोले और माउस वाले गोले के बीच की दूरी के तौर पर रिटर्न (प्रदान) की है। माउस वाले गोले को इस प्रकार खिसकाएं कि वह बीच वाले गोले के ऊपर पहुंच जाए और देखें कि दूरी की कीमतें किस प्रकार बदलती हैं।



**इसके बारे में सोचें:** आपके विचार में दूरी की किस रेंज की कीमतें यह दिखाएंगी कि दोनों ने एक-दूसरे को छुआ है? हम खुद से यह प्रश्न बाद में करेंगे, जब हम हमारे प्रोजेक्ट रेखा-चित्र को पूरा करते समय एक कंडीशनल लिखेंगे ताकि हम “पकड़ने” का पता लगा सकें।

### अपने कोड में `dist()` जोड़ें (5-8 मिनट)

आइए हमारे प्रोजेक्ट रेखा-चित्र में `dist()` फंक्शन जोड़ते हैं। सबसे पहले हम फंक्शन द्वारा रिटर्न (प्रदान) की गई कीमत (यानि कैचर के केंद्र बिंदु और उल्का के केंद्र बिंदु के बीच की दूरी) को स्टोर करने के लिए एक चर राशि बनाएंगे। उसके बाद हम फंक्शन जोड़ेंगे। और आखिर में हम लगातार बदलती दूरी की कीमत का पता लगाने के लिए एक नए फंक्शन, `print()` फंक्शन का उपयोग करेंगे।

- ❑ **दूरी की कीमत स्टोर करने के लिए `setup()` के ऊपर (यानि उससे पहले) एक नया चर राशि जोड़ें।** हमने हमारे चर राशि को `distance` नाम दिया है, पर आप इसे जो चाहें वो नाम दे सकते हैं। बस इतना याद रखें कि आपको अपने बाकी के पूरे कोड में वही नाम प्रयोग करना है। आपको उसे कीमत निर्धारित करने की ज़रूरत नहीं है।
- ❑ **कैचर कोड के अंतर्गत `dist()` फंक्शन को कॉल करें।** यदि आपको सहयोगपूर्ण लगे तो आप खुद को यह याद दिलाने के लिए एक छोटा सा कमेंट जोड़ सकते हैं कि यह कोड की पंक्ति क्या करती है।
- ❑ `x1` और `y1` के पैरामीटर्स को उन चर राशियों पर सेट करें जो आपकी उल्का के `x` और `y` स्थान स्टोर करते हैं।
- ❑ `x2` और `y2` के पैरामीटर्स को `mouseX` और `mouseY` पर सेट करें।
- ❑ `dist()` फंक्शन के परिणामों को अपने `distance` चर राशि में स्टोर करें।



अपने कोड को पृष्ठ 4 पर संदर्भ मार्गदर्शिका से जांचना न भूलें।

## चरण 6: दूरी की गणना करना (जारी)

### dist() की कीमत को कंसोल पर प्रिंट करें (3-5 मिनट)

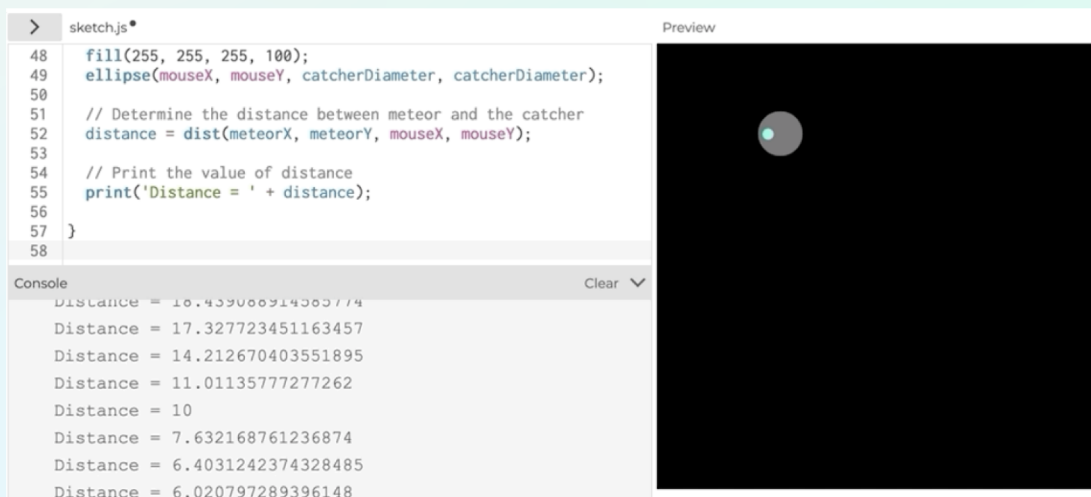
अब हमारा प्रोग्राम उल्का के केंद्र बिंदु और कैचर के केंद्र बिंदु के बीच की दूरी की लगातार गणना कर रहा है! पर हम वे कीमतें कैसे जांच सकते हैं? इसे करने का सबसे आसान तरीका है `print()` फंक्शन। यह एल्फान्यूमेरिकल कीमतें, यानि शब्द और संख्याएं, एडिटर के नीचे कंसोल पर प्रिंट करता है। यह रही इसकी वाक्य-रचना:

जावास्क्रिप्ट	विवरण
<pre>print('Distance = ' + distance);</pre>	<ul style="list-style-type: none"><li>→ <b>print</b>: उस फंक्शन का नाम जो संदेशों को कंसोल पर प्रिंट करता है।</li><li>→ <b>()</b>: हम कोष्ठकों द्वारा हमारे प्रोग्राम को यह बताते हैं कि उसे इस फंक्शन को कॉल करना है। कभी-कभी हम हमारे कोष्ठकों के अंदर फंक्शन के पैरामीटर या इनपुट शामिल कर देते हैं।</li><li>→ <b>' '</b>: सिंगल या डबल कोट्स प्रोग्राम को बताते हैं कि हम स्ट्रिंग या टेक्स्ट प्रिंट कर रहे हैं। आप दोनों से किसी भी प्रकार के कोट्स का उपयोग कर सकते हैं, पर जिससे ओपन करें उसी से क्लोज़ भी करें।</li><li>→ <b>+</b>: घटकों को एक साथ प्रिंट करने के लिए उन्हें जोड़ देता है।</li><li>→ <b>distance</b>: distance चर राशि में स्टोर वर्तमान कीमत को प्रिंट करता है।</li><li>→ <b>;</b>: p5.js की सभी कोड की पंक्तियों का अंत सेमीकोलन से होना आवश्यक होता है।</li></ul>

आइए इसे हमारे रेखा-चित्र में लिख देते हैं:

- ❑ distance चर राशि की कीमत प्रिंट करने के लिए यह कोड की पंक्ति `dist()` फंक्शन के नीचे जोड़ें:  
`print('Distance = ' + distance);`
- ❑ रेखा-चित्र चलाएं।

**Distance** = टेक्स्ट, और बदलती कीमतों की एक शृंखला कंसोल पर प्रिंट होनी चाहिए।



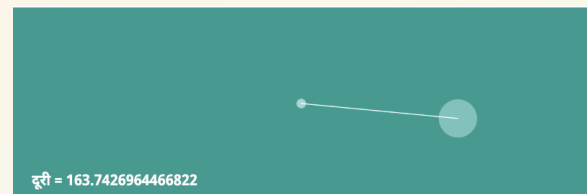


## चरण 7: स्पर्श का निर्धारण (5 मिनट)

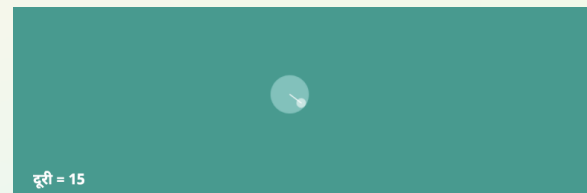
### दूरी की कीमत का उपयोग करके स्पर्श परिभाषित करें (2-3 मिनट)

हम जानते हैं कि हम चाहते हैं कि कैचर और उल्का के एक-दूसरे को छू लेने पर हमारा प्रोग्राम एक निश्चित ढंग से व्यवहार करे। इसे करने के लिए, हमें एक न्यूमेरिकल (संख्यात्मक) कीमत चाहिए जो यह स्पर्श दिखाती हो। हम **distance** (याद रखें कि हो सकता है कि आपने चर राशि का कोई अन्य नाम रखा हो) में स्टोर हुई कीमत का उपयोग करके पता कर सकते हैं कि कब कैचर और उल्का एक-दूसरे को छूते हैं।

पूर्व में आपने जो **दूरी डेमो रेखा-चित्र** देखा था उस पर दोबारा विचार करें: जब कैचर “उल्का” को पूरी तरह या उसके अधिकतर भाग को ढक लेता है तो दूरी की कीमत क्या होती है?



डेमो रेखा-चित्र देखने के बाद, हम यह कह सकते हैं कि **यदि दूरी 15 से कम है** तो स्पर्श हुआ है। अब हम हमारे प्रोग्राम में कंडीशनल्स का उपयोग करके कुछ निर्णय लेना शुरू कर सकते हैं।



### कंडीशनल्स की पुनरावृत्ति (2 मिनट)

अब हम हमारे रेखा-चित्र को बता सकते हैं कि स्पर्श होने पर क्या करना है। इसके लिए हमें एक कंडीशनल स्टेटमेंट (सशर्त कथन) चाहिए होगा। आइए एक छोटी सी पुनरावृत्ति कर लेते हैं: **कंडीशनल स्टेटमेंट (सशर्त कथन)** हमें हमारे प्रोग्राम के प्रवाह (फ्लो) का नियंत्रण करने की सुविधा देते हैं। वे **if** स्टेटमेंट का उपयोग करके जांचते हैं कि कोई कंडीशन (शर्त) **true** (सत्य) है या **false** (असत्य)। यदि कंडीशन टू (सत्य) है तो कंप्यूटर **if** स्टेटमेंट के अंदर मौजूद कोड चलाएगा।

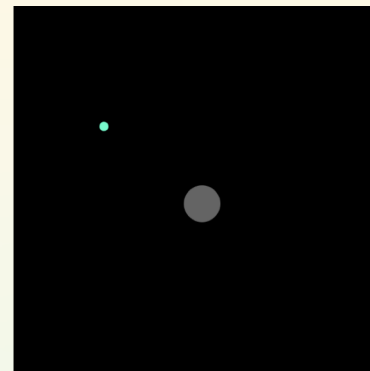
नीचे जावास्क्रिप्ट में कंडीशनल स्टेटमेंट की वाक्य-रचना स्पष्ट करने वाला एक उदाहरण दिया जा रहा है (याद करें कि p5.js जावास्क्रिप्ट की एक लाइब्रेरी है)। यह हमारे प्रोग्राम से कहता है कि यदि माउस का x स्थान 100 से अधिक है तो वह कैनवास के ऊपरी बायें कोने में एक गोला बना दे।

जावास्क्रिप्ट	विवरण
<pre>if(mouseX &gt; 100){   ellipse(0,0,50,50); }</pre>	<ul style="list-style-type: none"><li>→ <b>if</b>: प्रोग्राम को यह बताने वाला कीवर्ड (मुख्य शब्द) कि यह एक कंडीशनल है।</li><li>→ <b>()</b>: हम कोष्ठकों का उपयोग करके हमारे प्रोग्राम को यह बताते हैं कि उसके अंदर जो भी कुछ है वह उस कंडीशन (शर्त) का हिस्सा है जिसे वह जांचेगा।</li><li>→ <b>mouseX &gt; 100</b>: वह कंडीशनल एक्सप्रेशन (व्यंजक) जिसे परख कर रेखा-चित्र जात करेगा कि वह टू (सत्य) है या फ़ॉल्स (असत्य)। आप <b>तुलना संक्रियकों (कंपेरिजन ऑपरेटर्स)</b> जैसे <b>&gt;</b> (से बड़ा) या <b>&lt;=</b> (से कम या बराबर) या <b>तर्क संक्रियकों (लॉजिकल ऑपरेटर्स)</b> जैसे <b>  </b> (or) या <b>&amp;&amp;</b> (and) का उपयोग करके अपना स्टेटमेंट बना सकते हैं।</li><li>→ <b>{}</b>: मझले कोष्ठक (कली ब्रेकेट्स) हमारे प्रोग्राम को बताते हैं कि यदि कंडीशन टू होती है तो कौनसी कोड की पंक्तियाँ चलानी हैं।</li><li>→ <b>ellipse(0,0,50,50)</b>: कंडीशन के टू होने की स्थिति में यह स्टेटमेंट निष्पादित होगा। आप यहां अन्य if स्टेटमेंट्स भी शामिल कर सकते हैं।</li><li>→ <b>;</b>: p5.js की सभी कोड की पंक्तियों का अंत सेमीकोलन से होना आवश्यक होता है।</li></ul>

## चरण 8: कैचर कंडीशनल बनाएं (5-10 मिनट)

### कैचर कंडीशनल की योजना बनाएं (2-4 मिनट)

हमारा पहला कंडीशनल, हमारे कैचर के लिए होगा। हमें एक ऐसा `if` स्टेटमेंट लिखना होगा जो परखेगा कि उल्का और कैचर ने एक-दूसरे को छुआ है या नहीं। कोई भी कोड लिखने से पहले, आइए वर्णन करें कि हम क्या होते देखना चाहते हैं। गेम दोबारा खेलें, और फिर कंडीशनल के लिए स्कूडोकोड (छद्म कोड) लिखें। जितना संभव हो उतना विशिष्ट बनने की कोशिश करें। शुरुआत के लिए आप वह स्कूडोकोड प्रयोग कर सकते हैं जो आपने **भाग 1 के योजना बनाने वाले अनुभाग** में लिखा था। शुरु करने से पहले [उदाहरण रेखा-चित्र](#) देखें।



**संकेत:** आप दूरी और उल्का के `y` स्थान के उन चर राशियों का उपयोग कर सकते हैं जिन्हें आपने पिछले चरणों में बनाया था। आपको स्पर्श दर्शाने वाली एक संख्या की भी ज़रूरत पड़ेगी।



अपने कोड को पृष्ठ 5 पर संदर्भ मार्गदर्शिका से जांचना न भूलें।

### कैचर कंडीशनल जोड़ें (3-5 मिनट)

अब, अपने स्कूडोकोड (छद्म कोड) को असली कोड में बदलें:

- ☐ एक `if` स्टेटमेंट जोड़ें जो `distance` चर राशि की कीमत जांचता हो।
- ☐ एक कोड की पंक्ति जोड़ें जो आपकी उल्का के `y` स्थान को अपडेट करती हो। सुनिश्चित करें कि वह कंडीशनल के मझले कोष्ठक (कर्ली ब्रेकेट्स) के अंदर हो!
- ☐ अपना कोड चलाएं।
- ☐ अपनी उल्का और कैचर का स्पर्श करवाने की कोशिश करें। जब वे स्पर्श करें, तो वर्तमान उल्का गायब हो जानी चाहिए और स्क्रीन पर सबसे ऊपर एक “नई” उल्का प्रकट होनी चाहिए।

यदि ऐसा नहीं होता है, तो कंसोल पर प्रिंट हो रही दूरी की कीमतें जांचें। जब कैचर और उल्का एक-दूसरे को स्पर्श करते हैं, तो क्या कीमतें, कंडीशनल स्टेटमेंट में निर्धारित की गई रेंज के अंदर होती हैं? यदि नहीं, तो आपको अपने एक्सप्लेन (व्यंजक) में संख्या में बदलाव करना होगा।

`print()` फंक्शन डीबगिंग में बहुत सहायक होता है! यदि किसी भी समय आप अपने रेखा-चित्र में किसी चर राशि की कीमत को लेकर अनिश्चित हों, तो उसे फिर से जांचने के लिए उसके नीचे एक `print()` स्टेटमेंट लिख दें।

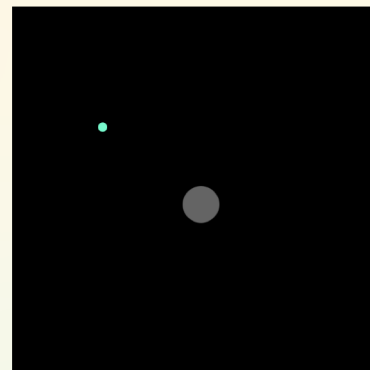


अपने कोड को पृष्ठ 5 पर संदर्भ मार्गदर्शिका से जांचना न भूलें।

## चरण 9: स्क्रीन के निचले भाग का कंडीशनल बनाएं (5-10 मिनट)

### स्क्रीन के निचले भाग के कंडीशनल की योजना बनाएं (2-3 मिनट)

अब चलिए यह परखने का कंडीशनल लिखते हैं कि उल्का ने स्क्रीन के निचले भाग को छुआ या नहीं। यदि उसने कैनवास के निचले भाग को छुआ है, तो हम चाहते हैं कि प्रोग्राम उसे कैनवास के ऊपरी भाग में दोबारा से बना दे। यदि आप अपने कोड में कैनवास की ऊंचाई या चौड़ाई के उपयोग का एक तेज़ तरीका चाहते हों, तो आप न्यूमेरिकल (संख्यात्मक) कीमत के स्थान पर `height` और `width` कीवर्ड्स (मुख्य शब्दों) का उपयोग कर सकते हैं। शुरु करने से पहले [उदाहरण रेखा-चित्र](#) देखें।



**याद रखें:** हमारे कैनवास की ऊंचाई 400 पिक्सल है।

इस कंडीशनल के लिए स्क्रिप्टकोड (छद्म कोड) लिखें। पहले की तरह, जितना संभव हो उतना विशिष्ट बनने की कोशिश करें। शुरुआत के लिए आप वह स्क्रिप्टकोड प्रयोग कर सकते हैं जो आपने **पिछले चरण** में या **भाग 1 के योजना बनाने वाले अनुभाग** में लिखा था।



अपने कोड को पृष्ठ 5 पर संदर्भ मार्गदर्शिका से जांचना न भूलें।

### स्क्रीन के निचले भाग का कंडीशनल जोड़ें (3-5 मिनट)

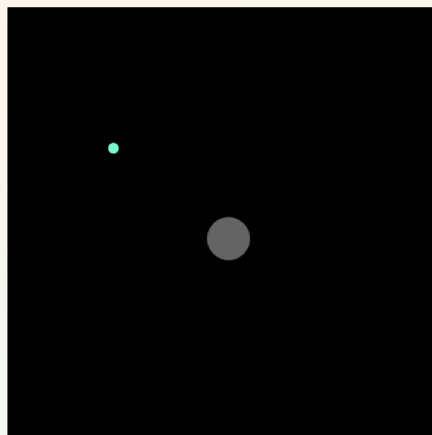
अब चलिए अपने स्क्रिप्टकोड (छद्म कोड) को असली कोड में बदलें:

- ☐ `if` स्टेटमेंट और कंडीशनल एक्सप्रेशन (सशर्त व्यंजक) लिखें।
- ☐ वह स्टेटमेंट जोड़ें जो कंडीशनल के टू (सत्य) होने की स्थिति में चलेगा। सुनिश्चित करें कि वह कंडीशनल के मझले कोष्ठक (कर्ली ब्रेकेट्स) के अंदर हो।
- ☐ अपना कोड चलाएं।
- ☐ उल्का के आपके कैनवास के निचले भाग से छूने का इंतज़ार करें। जब वे स्पर्श करें, तो वर्तमान उल्का गायब हो जानी चाहिए और स्क्रीन पर सबसे ऊपर एक “नई” उल्का प्रकट होनी चाहिए।

## चरण 10: अपने कोड को परखें (5-10 मिनट)

चलिए अब तक हमने जो लिखा है उसे परख कर यह सुनिश्चित करते हैं कि हमारा प्रोग्राम उसी तरह चल रहा हो जैसे हम चाहते हैं। प्ले बटन पर क्लिक करके अपना रेखा-चित्र चलाएं। इसे यह करना चाहिए:

- दूरी की कीमत को कंसोल पर प्रिंट करना।
- कैचर और उल्का का स्पर्श होने पर कैनवास के ऊपरी भाग में उल्का दोबारा से बनाना।
- उल्का और कैनवास के निचले भाग का स्पर्श होने पर कैनवास के ऊपरी भाग में उल्का दोबारा से बनाना।



उदाहरण रेखा-चित्र चलाने के लिए [यहां](#) क्लिक करें।

जैसा चाहा था वैसे कार्य नहीं कर रहा? ये डीबगिंग के सुझाव आजमाएं:

- क्या आपका कोड सही मझले कोष्ठकों (कर्ली ब्रैकेट्स) के अंदर है?
- क्या आपने हरेक कोड की पंक्ति के अंत में सेमीकोलन टाइप किए हैं?
- क्या आपने चर राशियों और फंक्शन के नामों की सही स्पेलिंग लिखी?
- क्या आपके फंक्शन्स सही स्थान पर और सही क्रम में हैं? याद रखें कि प्रोग्राम फ़्लो में क्रम मायने रखता है!
- क्या आपके `dist()` फंक्शन में मौजूद पैरामीटर्स सही हैं?
- `print()` का उपयोग करके कीमतों को कंसोल पर प्रिंट करें और उसके बाद जो भी `if` स्टेटमेंट्स हों उनकी जांच करें।



अपने कोड को पृष्ठ 6 पर संदर्भ मार्गदर्शिका से जांचना न भूलें।

## चरण 11: समझ की जाँच करें

मान लें कि आप चाहते हैं कि कैचर और उल्का के बाहरी किनारे का ज़रा सा स्पर्श होते ही आप उल्का को “पकड़” लें। आप अपने पहले कंडीशनल स्टेटमेंट के एक्सप्रेशन में कीमत को घटाएंगे या बढ़ाएंगे?



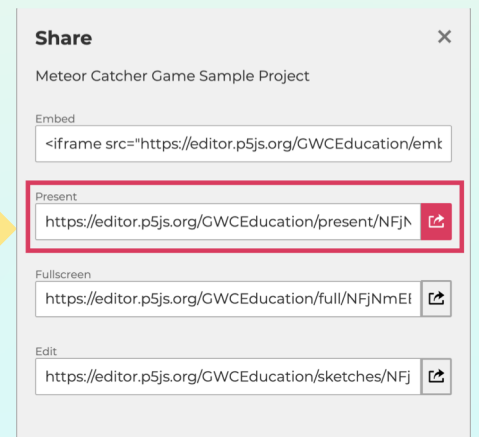
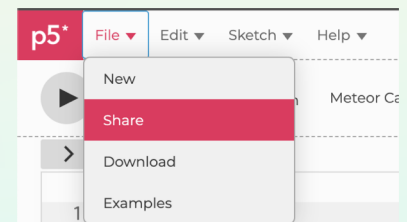
अपने विचारों को पृष्ठ 7 पर संदर्भ मार्गदर्शिका से जांचना न भूलें।

## चरण 12: अपने Girls Who Code at Home परियोजना को साझा करें! (5 मिनट)

हम आपके काम को देखना पसंद करेंगे और हम जानते हैं कि दूसरे भी ऐसा करेंगे। अपना गेम हमसे साझा करें! @girlswhocode #codefromhome को टैग करना मत भूलें, और हो सकता है कि हम आपको हमारे खाते में प्रदर्शित कर देंगे!

अपनी परियोजना को साझा करने के लिए निम्नलिखित चरणों का अनुसरण करें:

- पहले अपनी परियोजना को सहेजें।
- **फ़ाइल (File)** मेन्यू में, ड्रॉपडाउन मेन्यू में से **शेयर (Share)** विकल्प चुनें।
- ड्रॉपडाउन मेनू में **Link** विकल्प को चुनें।
- **वर्तमान** लिंक को कॉपी करें और उसे जहाँ कहीं भी आप साझा करना चाहती हैं वहाँ पेस्ट करें।



परियोजना का लिंक

और Girls Who Code at Home परियोजनाओं के लिए बनी रहें!

