



Girls Who Code At Home

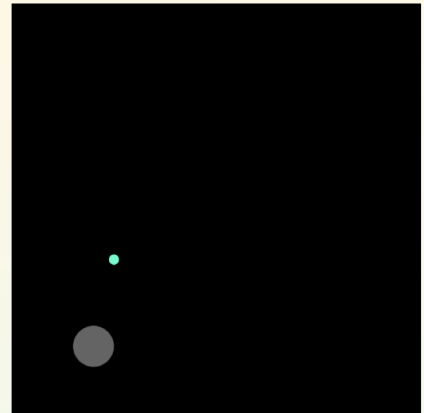
उल्का पकड़ो गेम: भाग 5

एक रैंडमाइज़र जोड़ें

गतिविधि अवलोकन

इस अंतिम भाग में, हम आपके गेम को अधिक चुनौतीपूर्ण - और अधिक मजेदार - बनाने का एक तरीका देखेंगे! आप सीखेंगे कि कैसे आप [random\(\)](#) फंक्शन का उपयोग करके अपनी उल्का के व्यवहार में जटिलता ला सकते हैं। अंत में, हमारे एक या अधिक एक्सटेंशन्स को आजमाकर अपने प्रोजेक्ट को आवश्यकतानुसार बदलें। गतिविधि के अंत तक पहुंचने पर आप जो कुछ सीखेंगे उसके पूर्ववलोकन 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](#), [भाग 3](#), और [भाग 4](#) पूरा कर लिया हो।



सीखने के लक्ष्य

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

- यह समझना कि गेम की चुनौती को बढ़ाने या घटाने के लिए [random\(\)](#) फंक्शन का किस प्रकार उपयोग किया जा सकता है।

सामग्रियां

- [p5.js ऑनलाइन एडिटर](#)
- [उल्का पकड़ो गेम सैंपल प्रोजेक्ट](#)
- [उल्का पकड़ो गेम भाग 5 संदर्भ मार्गदर्शिका](#)



वुमन इन टेक स्पॉटलाइट: लिसेट टाइट-मॉन्टगोमरी (Lisette Titre-Montgomery)



तस्वीर स्रोत:

[लिसेट टाइट-मॉन्टगोमरी](#)

(Lisette Titre-Montgomery)

जब आप डैंस सेंटरल (Dance Central) में थिरकते हैं या द सिम्स (The Sims) में किरदार बनाते हैं, तो क्या आपने कभी आपको वहां ले जाने वाली टेक्नॉलजी के बारे में सोचा है? लिसेट टाइट-मॉन्टगोमरी (Lisette Titre-Montgomery) मार्केट के सबसे लोकप्रिय गेम्स के पीछे के इंजीनियर और डिज़ाइनर के रूप में कार्य करती हैं। गेम उद्योग में सत्रह वर्ष से अधिक के अनुभव और मार्केट में उतारे जा चुके तेरह गेम्स के साथ, उन्होंने वीडियो गेमर और डेवलपर के बीच के फासले को सफलतापूर्वक भर दिया है।

लिसेट वाइट हाउस के साथ परामर्श और समावेशी भर्ती पहलों के जरिए गेमिंग उद्योग में अधिक विविधता लाने में भी अपना समय लगाती हैं। वे ओकलैंड, कैलिफ़ोर्निया स्थित [गेमहेड्स \(Gameheads\)](#) नामक एक प्रोग्राम के साथ छात्राओं को स्टीम (STEAM) शिक्षा एवं करियर में संलग्न करने के लिए K-12 शिक्षा में गेम आधारित पाठ्यचर्याओं की निरंतर पक्षधरता करती हैं।

लिसेट के बारे में, विभिन्न गेम्स पर काम करते समय उनके सामने आई विभिन्न चुनौतियों के बारे में, और उनसे जीतने के उनके तरीके के बारे में और जानने के लिए यह [वीडियो](#) देखें।

लिसेट और उनके कार्य के बारे में और जानना चाहते हैं?

- उनकी [व्यक्तिगत वेबसाइट](#) देखें और उनके कुछ काय और प्रोजेक्ट्स के बारे में और जानें।
- लिसेट की पृष्ठभूमि और वे गेमिंग उद्योग में कैसे आई इस बारे में यह [लेख](#) पढ़ें।
- उनके करियर के कुछ मुख्य बिंदु देखने के लिए, और गेमिंग उद्योग में शुरुआत की इच्छा रखने वाली छात्राओं को उनकी क्या सलाह है यह जानने के लिए [लिसेट की प्रोफाइल](#) देखें!

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 लिसेट 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।



वीरता

लिसेट ने अपनी पहली जॉब से पहले गेमिंग उद्योग में कभी काम नहीं किया था। अपनी भूमिका में सफल होने के लिए उन्होंने कौनसे कदम उठाए?

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

चरण 1: अपने गेम की चुनौती पर चिंतन करें (2-4 मिनट)

इस समय, हमने हमारी गेम के सभी महत्वपूर्ण भाग कवर कर लिए हैं: घटक, मूल यांत्रिकी, गेम और स्थान। पर अभी तक गेम ज़्यादा मज़ेदार नहीं बना है, क्योंकि यह पर्याप्त रूप से चुनौतीपूर्ण नहीं है। कठिनाई का सही स्तर ढूंढना, गेम डिज़ाइनर्स के लिए सबसे महत्वपूर्ण होता है। कठिनाई बढ़ाने के लिए आप जिन कई सारे टूल्स का उपयोग कर सकते हैं उनमें से एक टूल है रैंडमनेस यानि संयोगाधारिता को शामिल कर देना। इससे खिलाड़ी यह पूर्वानुमान नहीं लगा पाता कि आगे क्या आने वाला है, और वह नई जानकारी से चकित हो जाता है जिसके प्रति उसे खुद को ढालना ही पड़ता है।

गेम को अधिक चुनौतीपूर्ण बनाने के लिए हम रैंडमनेस कहां जोड़ सकते हैं? इस बारे में एक मिनट सोचें, और फिर अपने विचारों की तुलना नीचे दिए गए विचारों से करें।



अपने विचारों को पृष्ठ 2 पर संदर्भ मार्गदर्शिका से जांचना न भूलें।

चरण 2: random() फंक्शन को जानें (5 मिनट)

हम random() फंक्शन का उपयोग करके हमारे कोड में रैंडमनेस जोड़ सकते हैं। यह फंक्शन एक रैंडम प्लोटिंग पॉइंट संख्या रिटर्न (प्रदान) या आउटपुट प्रदान करता है जो कीमतों की एक निर्धारित रेंज के अंदर होता है। प्लोटिंग पॉइंट का यह अर्थ है कि संख्या में दशमलव हो सकता है। इससे कहीं अधिक संख्याओं का उपयोग संभव हो जाता है। यह [उदाहरण रेखा-चित्र](#) देखें जिसमें कैनवास के अंदर हर माउस क्लिक से fill() फंक्शन में रेड, ग्रीन और ब्लू कीमतों के लिए एक रैंडम कीमत उत्पन्न होती है।



चलिए random() फंक्शन का वाक्य-विन्यास देखते हैं।

जावास्क्रिप्ट	विवरण
<code>random(min, max);</code>	<ul style="list-style-type: none">→ random: फंक्शन का नाम।→ (): 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।→ min: आपकी चाही रैंडम रेंज का निचला सिरा जिसमें यह संख्या एक विकल्प के रूप में शामिल रहती है।→ max: आपकी चाही रैंडम रेंज का ऊपरी सिरा जिसमें यह संख्या एक विकल्प के रूप में ये शामिल नहीं रहती है।→ ;: जावास्क्रिप्ट की सभी कोड की पंक्तियों का अंत सेमीकोलन से होना आवश्यक होता है।

चरण 3: अपने स्यूडोकोड (छद्म कोड) को अपडेट करें (2-4 मिनट)

इससे पहले कि हम हमारी नई कोड की पंक्तियाँ जोड़ें, आइए पहले इंसानी भाषा में इस बारे में थोड़ा मंथन करें कि हम क्या करवाना चाहते हैं। हर `random()` फंक्शन के लिए स्यूडोकोड (छद्म कोड) लिखें। अधिकतम संभव विशिष्ट होने की कोशिश करें और निर्धारित कीमत रेंज का उपयोग करें। जब आपका कार्य पूरा हो जाए, तो नीचे अपने उत्तर की जांच करें:

- **उल्का का आरंभ स्थान।** कीमत की रेंज: 0 से 400
- **उल्का की चाल (स्पीड)।** कीमत की रेंज: 0.5 से 4
- **उल्का का आकार।** कीमत की रेंज: 10 से 30



अपने कोड को पृष्ठ 2 पर संदर्भ मार्गदर्शिका से जांचना न भूलें।

चरण 4: अपना कोड जोड़ें (5-8 मिनट)

अब हमें आपके स्यूडोकोड (छद्म कोड) को असली कोड में बदलना है। हम जो कोड लिखेंगे वह सारा-का-सारा मौजूदा स्टेटमेंट के नीचे दोनों कंडीशनल स्टेटमेंट्स के अंदर जाएगा। इस समय, हम उल्का का स्पर्श कैचर या नीचे वाली दीवार से होने पर उसे स्क्रीन के ऊपरी भाग में दोबारा बना मात्र रहे हैं। कंडीशनल में ये स्टेटमेंट्स जोड़ने से नई उल्का का स्थान, उसकी चाल (स्पीड) और उसका आकार, ये सभी चीज़ें रैंडम हो जाएंगी।

पहला कंडीशनल स्टेटमेंट ढूंढें। ये नई कोड की पंक्तियाँ `meteorY = 0;` के नीचे जोड़ें:

- ☐ वह कोड की पंक्ति लिखें जो उल्का के आरंभ स्थान को 0 से 400 के बीच की कोई भी रैंडम कीमत देती है।
- ☐ वह कोड की पंक्ति लिखें जो उल्का की चाल को 0.5 से 4 के बीच की कोई भी रैंडम कीमत देती है।
- ☐ वह कोड की पंक्ति लिखें जो उल्का की आकार को 10 से 30 के बीच की कोई भी रैंडम कीमत देती है।

दूसरा कंडीशनल स्टेटमेंट ढूंढें।

- ☐ पहले कंडीशनल के लिए आपने अभी-अभी जो स्टेटमेंट्स लिखे थे उन्हें कॉपी करें, और दूसरे कंडीशनल में `meteorY = 0;` के नीचे पेस्ट कर दें।

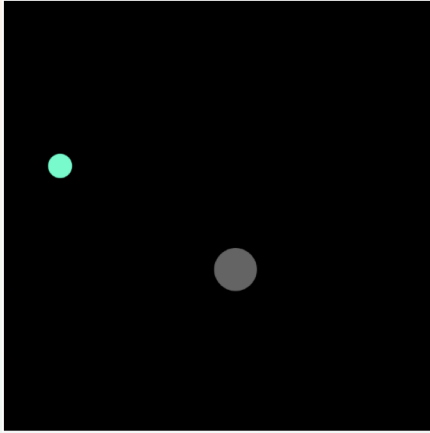
शायद आपका ध्यान इस ओर गया हो कि कंडीशनल्स के अंदर जो कोड की पंक्तियाँ हैं वे एक जैसी दिखती हैं! सही बात है! आप इन कंडीशनल्स को **तर्क संचालक (लॉजिकल ऑपरेटर)** "or" - `||` का उपयोग करके संयुक्त कर सकते हैं। आप उन्हें एक साथ 'नेस्ट' करके (एक के अंदर दूसरा रख कर) भी संयुक्त कर सकते हैं। हमने उन्हें अलग करने का विकल्प इसलिए चुना ताकि एक्सटेंशन्स की भ्रामकता थोड़ी घट जाए। यदि आप इन कंडीशनल्स को संयुक्त करना चाहें, तो आपका स्वागत है!

चरण 5: 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।

अपना कोड चलाएं और निम्नलिखित को परखें:

- ❑ अपनी उल्का और कैचर का स्पर्श करवाने की कोशिश करें। जब वे स्पर्श करें, तो वर्तमान उल्का गायब हो जानी चाहिए और एक रैंडम x स्थान पर नए आकार और नई चाल (स्पीड) वाली एक “नई” उल्का प्रकट होनी चाहिए।
- ❑ उल्का के आपके कैनवास के निचले भाग से छूने का इंतज़ार करें। जब वे स्पर्श करें, तो वर्तमान उल्का गायब हो जानी चाहिए और एक रैंडम x स्थान पर नए आकार और नई चाल (स्पीड) वाली एक “नई” उल्का प्रकट होनी चाहिए।



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 (कलीं ब्रैकेट्स) के अंदर है?
- क्या आपने हरेक कोड की पंक्ति के अंत में सेमीकोलन टाइप किए हैं?
- क्या आपके फंक्शन्स सही स्थान पर हैं? याद रखें कि प्रोग्राम फ़्लो में क्रम मायने रखता है!
- क्या आपके `random()` फंक्शन में मौजूद पैरामीटर्स सही हैं?



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

चरण 7: अपना कोड परखें और फीडबैक प्राप्त करें (5-10 मिनट)

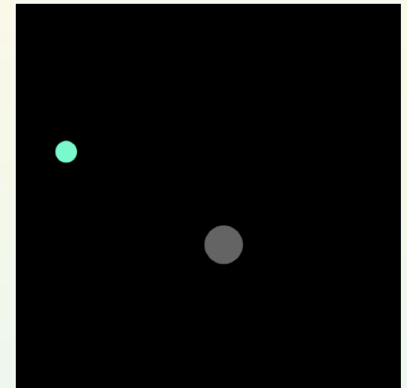


आप एक प्रमुख पड़ाव पर पहुंच गए हैं: आपने अपने प्रोजेक्ट के मूल भाग की रचना पूरी कर ली है! आगे बढ़ने से पहले, हम आपके कोड को परख कर यह सुनिश्चित करेंगे कि आपका गेम ठीक से कार्य कर रहा हो। यहां आप दोस्तों और परिजनों से अपने गेम पर फीडबैक भी मांग सकते हैं। अपना प्रोजेक्ट और यह चेकलिस्ट अपने पारखी (टेस्टर) से साझा करें!

चेकलिस्ट

अपना रेखा-चित्र चलाएं, और फिर नीचे वाली चेकलिस्ट का उपयोग करके अपना कोड परखें:

- ☐ आपके पास एक 400 गुणा 400 का कैनवास है। बैकग्राउंड में आपकी पसंद का रंग भरा हुआ है।
- ☐ आपके गेम की शुरुआत एक उल्का (यानि एक दीर्घवृत्त) के कैनवास के ऊपरी मध्य बिंदु से धीरे-धीरे नीचे की ओर गिरने से होती है। उल्का की कोई रूपरेखा नहीं है और उसमें आपकी पसंद का रंग भरा है।
- ☐ गेम में एक कैचर (यानि एक दीर्घवृत्त) है जो सफेद रंग का है और अर्द्ध-पारदर्शी है। कैचर का केंद्र बिंदु आपके माउस के साथ-साथ चलता है।
- ☐ आपका प्रोग्राम उल्का और कैचर के बीच की दूरी कंसोल पर प्रिंट करता है।
- ☐ यदि उल्का और कैचर एक-दूसरे को छूते हैं तो गेम अपडेट होता है और एक नई उल्का बनाता है।
- ☐ यदि उल्का, कैनवास के निचले भाग को छू जाती है, तो गेम अपडेट होता है और एक नई उल्का बनाता है।
- ☐ नई उल्का कैनवास के ऊपरी भाग में x अक्ष पर एक रैंडम स्थान पर एक रैंडम चाल (स्पीड) और रैंडम आकार के साथ शुरुआत करती है।



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

डीबगिंग के सुझाव

यदि आपका रेखा-चित्र उस तरह कार्य नहीं कर रहा जैसे आप चाहते हैं, तो सबसे पहले कंसोल को जांच कर देखें कि कहीं कोई त्रुटि तो नहीं है। आप अपने कोड को ठीक करने की कोशिश कर सकते हैं या उत्तर ढूंढने के लिए Google को आजमा सकते हैं। आप p5.js के [इस संसाधन](#) की भी मदद ले सकते हैं।

- 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 राशीयों और फंक्शन के नामों की सही स्पेलिंग लिखी?
- क्या आपके फंक्शन्स सही स्थान पर और सही क्रम में है? याद रखें कि प्रोग्राम फ़्लो में क्रम मायने रखता है!
- क्या आपके फंक्शन्स में मौजूद पैरामीटर्स कीमतें सही हैं? विशेष रूप से, `dist()` और `random()` फंक्शन्स की?
- `print()` का उपयोग करके वैल्यूज़ को कंसोल पर प्रिंट करें और उसके बाद जो भी `if` स्टेटमेंट्स हों उनकी जांच करें।

आपने कर दिखाया! आपने अपने प्रोजेक्ट की बुनियाद पूरी कर ली है। इस गतिविधि के अगले भाग में आपको अपने प्रोजेक्ट को कुछ वैकल्पिक एक्सटेंशन्स की मदद से और भी अधिक पर्सनलाइज़ करने का मौका मिलेगा।



चरण 8: विस्तार (5-45 मिनट)

एक्सटेंशन 1: चित्रों से कहानी बदलें (15-45 मिनट)

इस समय, हमारा गेम अंतरिक्ष और उल्काओं के बारे में है - पर इसका ऐसा ही होना आवश्यक नहीं है! आप नए चित्र या छवियां जोड़ कर गेम की कहानी बदल सकते हैं। गेम डिज़ाइन में इसे स्किनिंग कहते हैं। घटकों के रंग-रूप और गेम के बैकग्राउंड को गेम की स्किन कहा जाता है। आप यांत्रिकी को बदले बिना गेम की स्किन बदल सकते हैं, पर इससे गेम के अर्थ में और गेम के प्रति खिलाड़ियों के एहसास में बड़ा बदलाव आ सकता है।



इस एक्सटेंशन का उदाहरण रेखा-चित्र देखने के लिए [यहां](#) क्लिक करें।

इस एक्सटेंशन में, एक या अधिक आकृतियों और बैकग्राउंड के स्थान पर इमेज रखने की कोशिश करें। इस एक्सटेंशन को पूरा करने के लिए आपको इन बुनियादी चरणों की ज़रूरत होगी:

- ❑ पारदर्शी बैकग्राउंड वाली **.png** इमेज फ़ाइल्स ढूंढें। आपको Google Draw में उनका आकार बदलने की ज़रूरत पड़ सकती है, जिसके बाद उन्हें एक **.png** इमेज फ़ाइल के रूप में एक्सपोर्ट कर लें।
- ❑ अपनी **.png** फ़ाइल्स को अपने रेखा-चित्र में अपलोड करें।
- ❑ इमेज फ़ाइल स्टोर करने के लिए एक चर राशि घोषित करें।
- ❑ **setup()** में **loadImage()** फंक्शन से हर इमेज चर राशि को आरंभ (इनीशियलाइज़) करें।
- ❑ **draw()** के अंदर **image()** फंक्शन का उपयोग करके स्क्रीन पर इमेज चित्रित करें। सही x और y स्थान चर राशियों का उपयोग करना न भूलें।
- ❑ बैकग्राउंड इमेज चर राशि को **background()** फंक्शन के जरिए पास करें।

एक्सटेंशन संसाधन

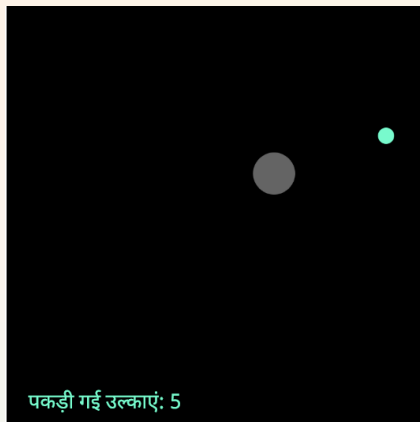
नीचे कुछ सहायक संसाधन दिए जा रहे हैं जिनका उपयोग करके हमने यह एक्सटेंशन बनाया था। इनसे आपको शुरुआत करने में मदद मिलेगी, पर याद रखें कि और भी बहुत से संसाधन मौजूद हैं जो आपसे बस एक सर्च इंजन की दूरी पर हैं!

- [p5.js वेब एडिटर: मीडिया फ़ाइल्स अपलोड करना - p5.js ट्यूटोरियल](#), द कोडिंग ट्रेन द्वारा (0:59 के आस-पास शुरू करें)
- [p5.js से एक इमेज उदाहरण लोड और प्रदर्शित करें](#) ध्यान दें: इस टेक्स्ट को अनदेखा कर दें जो कहता है कि 'To run this example locally, you will need an image file, and a running local server'. ऑनलाइन एडिटर के लिए यह लागू नहीं होता है।
- [loadImage\(\)](#)
- [image\(\)](#)
- [imageMode\(\)](#)
- [background\(\)](#)

[इस एक्सटेंशन के हमारे समाधान कोड](#) का एक लिंक यह रहा। हमारा सुझाव है कि सबसे पहले आप इसे खुद आजमाएं और इस संसाधन का उपयोग तब करें जब आप सच में अटक गए हों या अपने कार्य की जांच करना चाहते हों।

एक्सटेंशन 2: स्कोर पर नज़र रखें (5-15 मिनट)

ट्यूटोरियल में किसी बिंदु पर, शायद आपके मन में यह प्रश्न आया होगा कि, “मगर पॉइंट्स का क्या?!” इसे गेम बनाने के लिए खिलाड़ी के स्कोर पर नज़र रखना ज़रूरी तो नहीं है, पर इससे खिलाड़ियों को अपने खेल के बारे में तात्कालिक फीडबैक अवश्य मिल जाता है। आप स्कोर पर नज़र रखने के लिए एक चर राशि बना सकते हैं, और फिर जब भी खिलाड़ी पॉइंट अर्जित करे तो आप उस चर राशि की कीमत बढ़ा सकते हैं। आप यह भी कर सकते हैं कि खिलाड़ी की शुरुआत एक स्कोर के साथ हो, और जब वह कोई क्रिया विशिष्ट करे तो स्कोर की कीमत घट जाए।



इस एक्सटेंशन का उदाहरण रेखा-चित्र देखने के लिए [यहां](#) क्लिक करें।

इस एक्सटेंशन में, अपना एक साधारण सा स्कोरिंग सिस्टम बनाने की, और स्क्रीन पर स्कोर को टेक्स्ट के रूप में दर्शाने की कोशिश करें। इस एक्सटेंशन को पूरा करने के लिए आपको इन बुनियादी चरणों की ज़रूरत होगी:

- ❑ **स्कोर पर नज़र रखने के लिए एक चर राशि बनाएं और उसे शून्य पर सेट कर दें।** संकेत: यदि आप इस चर राशि को अपने रेखा-चित्र में कहीं से भी एक्सेस करना चाहते हैं, तो आपके विचार में आप उसे कहां रखना चाहेंगे?
- ❑ **यदि उल्का और कैचर का स्पर्श होता है तो इस चर राशि में वृद्धि करें।** संकेत: आप अपने कोड में कहां यह जांचते हैं कि उल्का और कैचर का स्पर्श हुआ या नहीं?
- ❑ **चर राशि की कीमत को स्क्रीन पर टेक्स्ट के रूप में चित्रित करें।**

एक्सटेंशन संसाधन

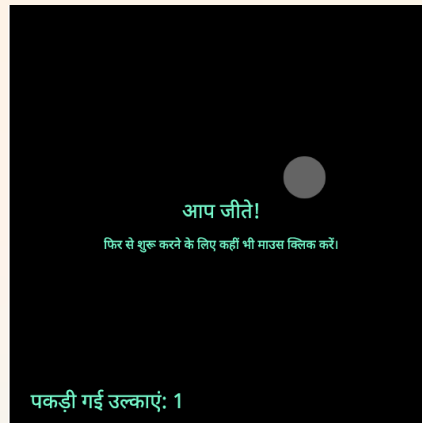
नीचे कुछ सहायक संसाधन दिए जा रहे हैं जिनका उपयोग करके हमने यह एक्सटेंशन बनाया था। इनसे आपको शुरुआत करने में मदद मिलेगी, पर याद रखें कि और भी बहुत से संसाधन मौजूद हैं जो आपसे बस एक सर्च इंजन की दूरी पर हैं!

- [p5.js से बढ़ाव घटाव उदाहरण](#)
- [p5.js से शब्द उदाहरण](#)
- [textSize\(\)](#)
- [text\(\)](#): आप *स्ट्रिंग्स* (यानि पाठ्य वण की शृंखला) को `text()` फंक्शन में चर राशियों के साथ संयुक्त कर सकते हैं।
- [textAlign\(\)](#)
- [string](#)

[इस एक्सटेंशन के हमारे समाधान कोड](#) का एक लिंक यह रहा। हमारा सुझाव है कि सबसे पहले आप इसे खुद आजमाएं और इस संसाधन का उपयोग तब करें जब आप सच में अटक गए हों या अपने कार्य की जांच करना चाहते हों।

एक्सटेंशन 3: विजय की अवस्था जोड़ें (10-20 मिनट)

चाहे अकेले खेलने के हों या साथ मिलकर, गेम्स में प्रायः एक विजय की अवस्था होती है। विजय की अवस्था तब आती है जब खिलाड़ी गेम की मुख्य चुनौती को सफलतापूर्वक पार कर लेते हैं, जैसे सबसे अधिक संख्या में गोल या 30 सेकंड में सबसे अधिक सोने के सिक्के या किसी अंतरिक्ष अभियान को पूरा करना। आप गेम को कई अलग-अलग तरीकों से जीत सकते हैं।



इस एक्सटेंशन का उदाहरण रेखा-चित्र देखने के लिए [यहां](#) क्लिक करें।

इस एक्सटेंशन में, यह परखने और खिलाड़ी को सूचित करने वाला एक सिस्टम बनाएं कि वह जीत गया है या नहीं, और फिर गेम को रीसेट करें ताकि खिलाड़ी गेम दोबारा खेल सके। शुरुआत करने से पहले, इस बारे में सोचें कि आपके गेम में “जीतने” का क्या अर्थ है। उदाहरण के लिए, क्या यह गेम कितनी बार खेला गया वह संख्या है, पॉइंट्स की संख्या है, आदि। [यहां](#) दिए गए एक्सटेंशन के उदाहरण में, हमने प्राप्त पॉइंट्स की संख्या का उपयोग यह तय करने के लिए किया है कि खिलाड़ी जीता या नहीं। यदि वह जीत जाता है और माउस क्लिक करता है, तो गेम रीस्टार्ट हो जाता है। इस एक्सटेंशन के लिए आपको अपना खुद का एक फंक्शन बनाना होगा, इसलिए यदि आप फंक्शन्स को परिभाषित करने के बारे में और जानना चाहते हैं, तो यह एक अच्छा मौका है!

इस एक्सटेंशन को पूरा करने के लिए आपको इन बुनियादी चरणों की ज़रूरत होगी:

- ❑ स्थान, स्कोर और चाल (स्पीड) की चर राशियों को उनकी मूल कीमतों पर रीसेट करके गेम को रीस्टार्ट करने का एक फंक्शन बनाएं।
- ❑ यह परखने का एक कंडीशनल बनाएं कि स्कोर, जीतने के लिए ज़रूरी पॉइंट्स की संख्या के बराबर है या नहीं।
- ❑ स्क्रीन पर एक संदेश लिखें जो खिलाड़ी को बताए कि वह जीत चुका है और गेम को रीस्टार्ट कैसे करना है।
- ❑ पहले कंडीशनल के अंदर एक दूसरा कंडीशनल बनाएं जो परखेगा कि माउस को क्लिक किया गया है या नहीं। उसके अंदर रीस्टार्ट फंक्शन को कॉल करें।

एक्सटेंशन संसाधन

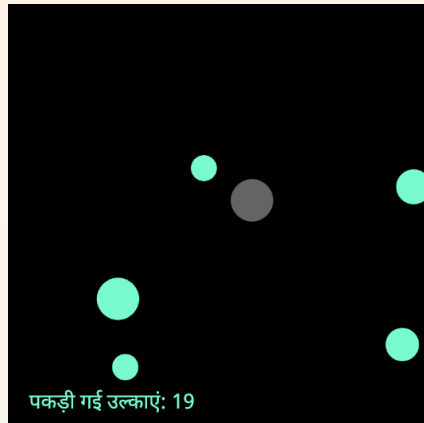
नीचे कुछ सहायक संसाधन दिए जा रहे हैं जिनका उपयोग करके हमने यह एक्सटेंशन बनाया था। इनसे आपको शुरुआत करने में मदद मिलेगी, पर याद रखें कि और भी बहुत से संसाधन मौजूद हैं जो आपसे बस एक सर्च इंजन की दूरी पर हैं!

- [फंक्शन की बुनियादी बातें - p5.js ट्यूटोरियल](#), द कोडिंग ट्रेन की ओर से वीडियो
- [फंक्शन](#)
- [mouseIsPressed](#)
- [p5.js से बढ़ाव घटाव उदाहरण](#)
- [p5.js से शब्द उदाहरण](#)
- [textSize\(\)](#)
- [text\(\)](#): आप *स्ट्रिंग्स* (यानी पाठ्य वण की शृंखला) को `text()` फंक्शन में चर राशियों के साथ संयुक्त कर सकते हैं।
- [textAlign\(\)](#)
- [string](#)

[इस एक्सटेंशन के हमारे समाधान कोड](#) का एक लिंक यह रहा। हमारा सुझाव है कि सबसे पहले आप इसे खुद आजमाएं और इस संसाधन का उपयोग तब करें जब आप सच में अटक गए हों या अपने कार्य की जांच करना चाहते हों।

एक्सटेंशन 4: और उल्काएं जोड़ें (25-40 मिनट)

निर्माण चरण के पिछले भाग में हमने देखा कि हम उल्का के गुणों को रैंडम बना कर किस तरह गेम को अधिक चुनौतीपूर्ण बना सकते हैं। गेम को अधिक चुनौतीपूर्ण बनाने का एक और तरीका यह है कि उल्काओं की संख्या बढ़ा दी जाए!



इस एक्सटेंशन का उदाहरण रेखा-चित्र देखने के लिए [यहां](#) क्लिक करें।

ध्यान दें: हमने एक स्कोर ट्रैकर भी शामिल किया है ताकि आप देख सकें कि कब उल्का स्पर्श करती है, पर इस एक्सटेंशन के लिए आपको उसे करना ज़रूरी नहीं है।

आप इसे कई तरीकों से कर सकते हैं।

- **तरीका #1:** एक दूसरी उल्का के लिए नई चर राशियां बनाएं। फिर पहली उल्का के लिए आपने जो कोड लिखा था उसकी कॉपी बना लें, पर उसमें दूसरी उल्का वाली चर राशियां अपडेट कर दें। यदि आपको लगता है कि यह तरीका थकाऊ है, तो आप सही सोच रहे हैं! इससे काम तो हो जाएगा, पर आपका कोड लंबा और बोझिल बन जाएगा।
- **तरीका #2:** [क्रम \(arrays\)](#) और [for](#) लूप्स का उपयोग करके और उल्काएं जोड़ें! क्रम (Arrays) से आपको एक अकेली चर राशि में घटकों (एलिमेंट्स) की एक क्रमबद्ध सूची स्टोर करने की सुविधा मिलती है। यानी, उल्का के व्यास की मात्र एक कीमत स्टोर करने की बजाय, आप 10, या 15, या 20 या 300 की कीमत स्टोर कर सकते हैं - और उन सभी को इंडेक्स नंबर का उपयोग करके एक्सेस कर सकते हैं। फ़ॉर (for) लूप्स आपको किसी कंडीशन (शर्त) के आधार पर कोड के किसी खंड को लूप करने (कई बार दोहराने) की सुविधा देते हैं। आप फ़ॉर (for) लूप्स और क्रम (arrays) को एक-दूसरे का परम मित्र कह सकते हैं क्योंकि आप [for](#) लूप का उपयोग करके [array](#) में स्टोर हर कीमत पर अपना कोड दोहरा सकते हैं। और जानने के लिए नीचे [एक्सटेंशन संसाधन](#) देखें।

इस एक्सटेंशन में, क्रम (arrays) और फ़ॉर (for) लूप्स का उपयोग करके उल्काओं की संख्या में चार की वृद्धि करके कुल पांच उल्काएं बनाने की कोशिश करें। क्रम (arrays) से आपको सारी उल्काओं की meteorX चर राशियां एक ही स्थान में स्टोर करने की सुविधा मिलेगी। फ़ॉर (for) लूप्स की मदद से आप हर उस चर राशि का बारी-बारी से उपयोग कर सकेंगे या उस पर कोई क्रिया कर सकेंगे।

इस एक्सटेंशन को पूरा करने के लिए आपको इन बुनियादी चरणों की ज़रूरत होगी:

- ❑ उल्का का X स्थान, उल्का का Y स्थान (ये सभी कीमतें 0 होनी चाहिए), उल्का का व्यास, दूरी और चाल (स्पीड) स्टोर करने के लिए **क्रम (arrays)** घोषित करें।
- ❑ **setup()** के अंदर **for** लूप बनाएं जो निम्नलिखित क्रम (arrays) में शुरूआती रैंडम कीमतें पहले से भरेंगे: X स्थान, उल्का का व्यास, और चाल (स्पीड)।
- ❑ एक **for** लूप बनाएं जो स्क्रीन पर उल्काएं चित्रित करता हो।
- ❑ एक **for** लूप बनाएं जो बारी-बारी से सभी उल्काओं पर काम करके उन्हें अलग-अलग चाल से नीचे गिराता हो।
- ❑ कैचर बनाएं।
- ❑ एक **for** लूप बनाएं जो बारी-बारी से सभी उल्काओं पर काम करके उल्का और कैचर के बीच की दूरी ज्ञात करता हो।
- ❑ **एक for लूप बनाएं जो बारी-बारी से सभी उल्काओं पर काम करके यह परखता हो कि किसी उल्का ने कैचर को छुआ या नहीं।** यदि उसने छुआ हो, तो उस उल्का को स्क्रीन के ऊपरी भाग में किसी रैंडम X स्थान पर दोबारा से बनाएं, और उस उल्का की एक नई रैंडम चाल (स्पीड) तय करें।
सुझाव: आपको for लूप के अंदर एक कंडीशनल स्टेटमेंट जोड़ना होगा।
- ❑ **एक for लूप बनाएं जो बारी-बारी से सभी उल्काओं पर काम करके यह परखता हो कि किसी उल्का ने स्क्रीन के निचले भाग को छुआ या नहीं।** यदि उसने छुआ हो, तो उस उल्का को स्क्रीन के ऊपरी भाग में किसी रैंडम X स्थान पर दोबारा से बनाएं, और उस उल्का की एक नई रैंडम चाल (स्पीड) तय करें।
सुझाव: आपको for लूप के अंदर एक कंडीशनल स्टेटमेंट जोड़ना होगा।

एक्सटेंशन संसाधन

नीचे कुछ सहायक संसाधन दिए जा रहे हैं जिनका उपयोग करके हमने यह एक्सटेंशन बनाया था। इनसे आपको शुरुआत करने में मदद मिलेगी, पर याद रखें कि और भी बहुत से संसाधन मौजूद हैं जो आपसे बस एक सर्च इंजन की दूरी पर हैं!

- द कोडिंग ट्रेन की ओर से p5.js के ट्यूटोरियल वीडियोज़
ध्यान दें: इनमें से कुछ वीडियोज़ ऑनलाइन एडिटर के बनने से पहले बनाए गए थे, पर वे उसी तरह काम करेंगे!
 - ◆ [क्रम \(array\) क्या होता है?](#)
 - ◆ [क्रम \(array\) और लूप](#)
 - ◆ [while और for लूप](#)
- [प्रोग्राम फ़्लो](#) ट्यूटोरियल
- [दोहराव \(इटरेशन\)](#) ट्यूटोरियल
- **for** यदि आपको शुरुआत करने में परेशानी हो रही है, तो पहले तरीके का उपयोग करके देखें और पहली उल्का के कोड की कॉपी करके दूसरी उल्का बनाएं। जब भी आपको कोई डबल (जैसे meteorY_1 और meteorY_2 या speed_1 और speed_2) दिखे तो इस बात की पूरी संभावना है कि आपको उन सभी के बारी-बारी से उपयोग के लिए क्रम (array) और फ़ॉर (for) लूप की ज़रूरत पड़ेगी।

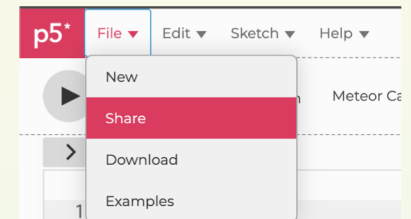
इस एक्सटेंशन के हमारे समाधान कोड का एक लिंक यह रहा। हमारा सुझाव है कि सबसे पहले आप इसे खुद आजमाएं और इस संसाधन का उपयोग तब करें जब आप सच में अटक गए हों या अपने कार्य की जांच करना चाहते हों।

चरण 9: अपने Girls Who Code at Home परियोजना को साझा करें! (5 मिनट)

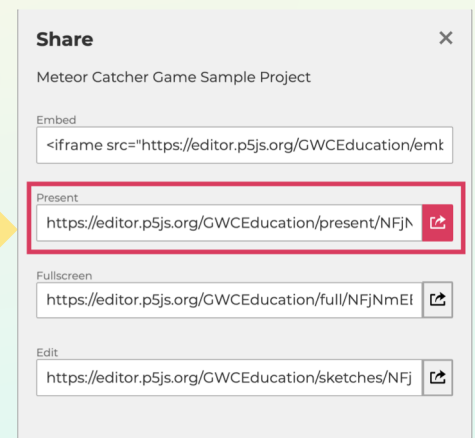
हम आपके काम को देखना पसंद करेंगे और हम जानते हैं कि दूसरे भी ऐसा करेंगे। अपना गेम हमसे साझा करें! @girlswhocode #codefromhome को टैग करना मत भूलें, और हो सकता है कि हम आपको हमारे खाते में प्रदर्शित कर देंगे!

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!

