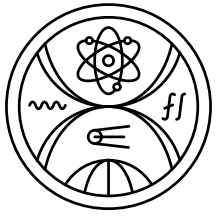
Orezávanie

Róbert Bohdal

robert.bohdal@fmph.uniba.sk
https:\\flurry.dg.fmph.uniba.sk\webog\bohdal-vyucba

Katedra algebry a geometrie
Fakulta matematiky fyziky a informatiky
Univerzita Komenského v Bratislave

Počítačová grafika (1)
prednáška č. 8



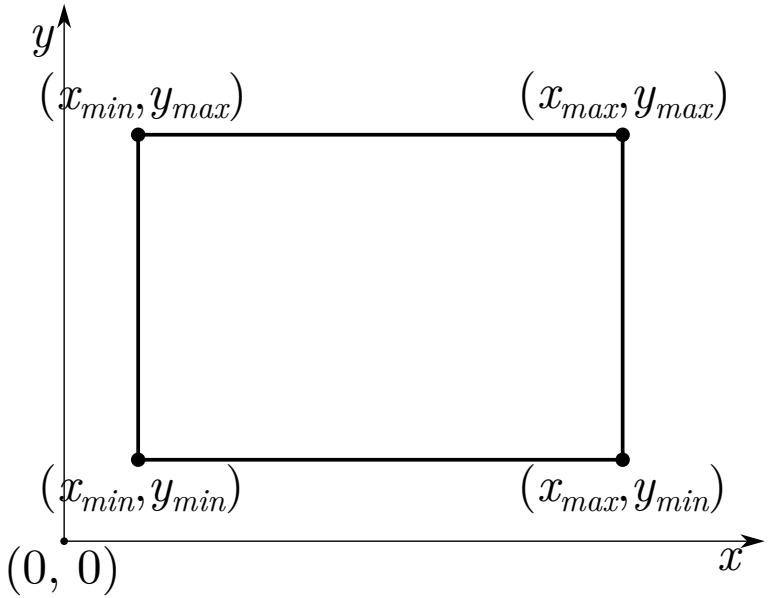
Obsah

Orezávanie

- Orezávaním rozumieme hľadanie takej časti objektu, ktorá je viditeľná v okne
- Orezávanie je špeciálny prípad prieniku objektu s oknom, ktoré sa vykonáva vždy, preto sa používajú efektívne algoritmy
- Štandardné okno má tvar axiálneho obdĺžnika, t.j. jeho hrany sú **rovnobežné** so súradnicovými osami a jeho vrcholy majú súradnice:

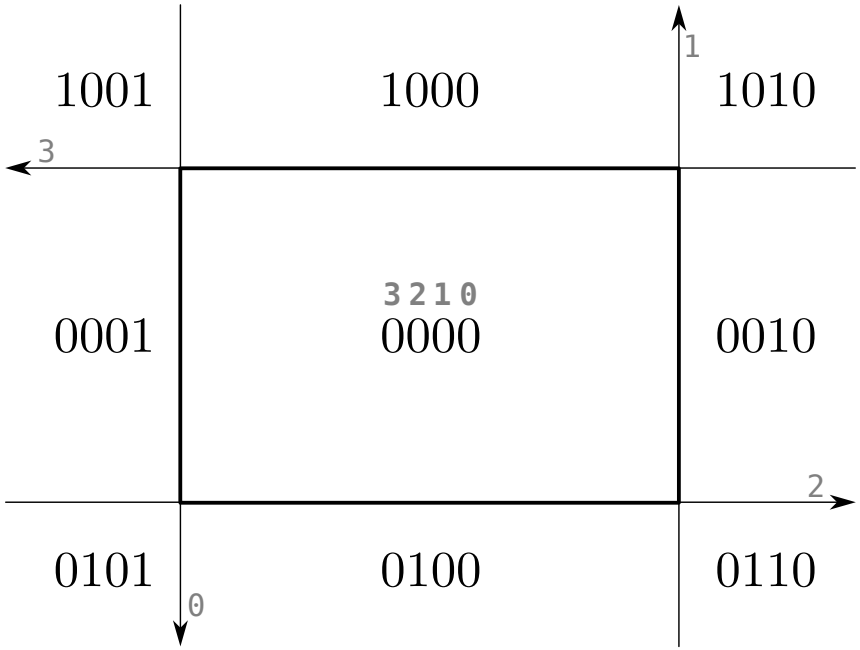
$$(x_{min}, y_{min}), (x_{max}, y_{min}), (x_{max}, y_{max}), (x_{min}, y_{max})$$

- Bod so súradnicami (x, y) leží v okne, ak $x_{min} \leq x \leq x_{max}$ a $y_{min} \leq y \leq y_{max}$
- Okno na zobrazovacej ploche má zvyčajne začiatok súradnicového systému vľavo hore, alebo vľavo dole



Algoritmus Cohen–Sutherland

- Tento algoritmus sa často vyskytuje v grafických knižniciach
- Priamky, na ktorých ležia strany okna rozdeľujú rovinu na 9 oblastí
- Každej z týchto oblastí je priradený štvorbitový kód, v ktorom konkrétny bit vyjadruje polpriestor určený konkrétnou orientovanou priamkou
- Koncovým bodom A, B orezávanej úsečky priradíme štvorbitový kód oblasti, v ktorej sa daný bod nachádza
- Overíme pomocou dvojice kódov priradených koncových bodov, či úsečku naozaj treba orezať

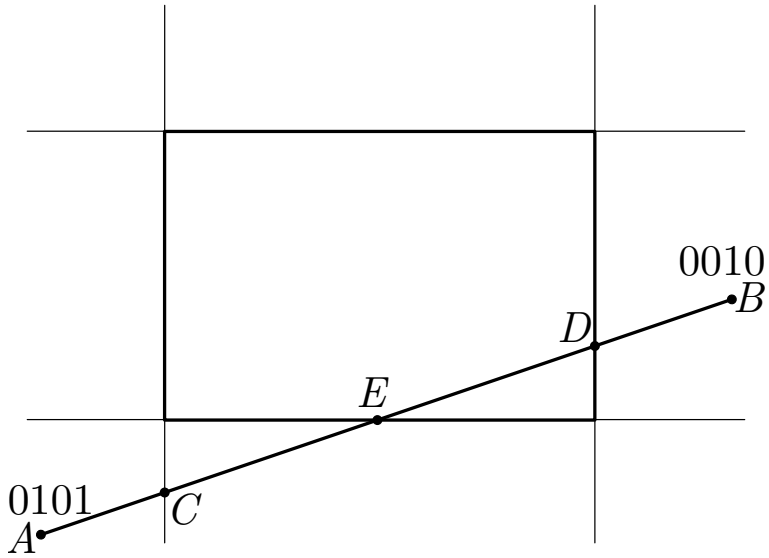


Algoritmus Cohen–Sutherland

1. Ak `kod(A)` = `kod(B)` = 0000, úsečku netreba orezať, keďže oba koncové body úsečky ležia **vnútri** okna
2. Ak `kod(A) && kod(B)` $\neq$ 0000, úsečku netreba orezávať, keďže oba koncové body úsečky ležia **naľavo**, **napravo**, **nad** alebo **pod** oknom
3. Ak nenastal ani jeden z dvoch predchádzajúcich prípadov, rozdelíme úsečku na dve podúsečky priamkou, ktorá leží na vybranej strane okna a každú z týchto podúsečiek potom **odznova orezávame** (t.j. priradíme kódy koncovým bodom a opäť testujeme)

- Operácia `&&` značí bitový súčin („and“) po bitoch, t.j. 0110 && 1010 = 0010

Príklad Algoritmus Cohen–Sutherland



1. Koncovým bodom A a B priradíme `kod(A)` $= 0101$, `kod(B)` $= 0010$
2. Kódy sú nenulové, teda koncové body úsečky ležia mimo okna
3. Keďže `kod(A) && kod(B)` $= 0000$, úsečku musíme rozdeliť na 2 časti. Pretože bod A leží naľavo od okna (najpravejší bit je 1) a bod B nie, rozdelíme úsečku priamkou ľavej hrany okna na úsečky AC a CB
4. Podľa kódov koncových bodov nových úsečiek vieme, že úsečka AC je pod oknom, preto ju vylúčime. Úsečku BC však musíme orezať ďalej
5. Bod B sa na rozdiel od bodu C nachádza vpravo od okna, preto príslušnou priamkou rozdelíme úsečku na CD a DB. Úsečku DB vylúčime, úsečku CD opäť rozdelíme na 2 časti a dostaneme orezanú úsečku ED

Modifikácie algoritmu Cohen–Sutherland

- Tento algoritmus je možné modifikovať tak, že keď úsečku treba rozdeliť, tak ju rozdelíme na 2 rovnaké časti, presne v strede
- Vzniknuté podúsečky znova rozdelíme, až kým o každej časti nevieme rozhodnúť, či leží celá vnútri alebo zvonka okna, prípadne kým nedosiahneme rozlíšenie výstupného zariadenia
- Výhodou tohto prístupu je, že netreba počítať prienik úsečky a priamky, avšak treba úsečku deliť na podúsečky mnohokrát
- Algoritmus sa dá veľmi ľahko modifikovať aj do 3D priestoru, v ktorom sa úsečka orezáva na axiálny kváder. V tomto prípade je treba ale použiť 6-bitové kódy

Algoritmus Cyrus–Beck (parametrické orezávanie)

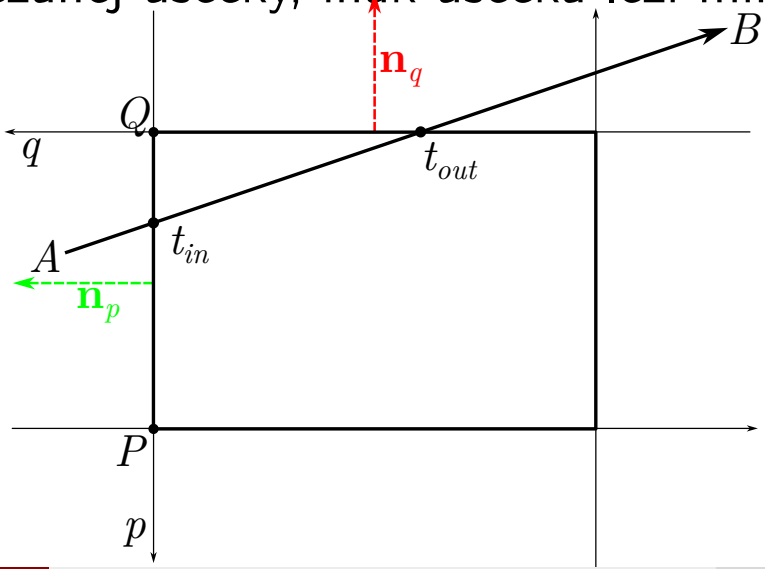
- Algoritmus používa parametrické vyjadrenie úsečky s koncovými bodmi $A = (a_x, a_y)$ a $B = (b_x, b_y)$:

$$x = a_x + \Delta x \cdot t$$
$$y = a_y + \Delta y \cdot t, \;\; t \in \langle 0, 1 \rangle ,$$

  kde $\Delta x = b_x - a_x$, $\Delta y = b_y - a_y$
- Predstavuje špeciálny prípad hľadania prieniku úsečky a **konvexného** mnohouholníka:
  1. pre každú priamku, prislúchajúcu danej strane okna vypočítame hodnotu parametra t_i, ktorý zodpovedá prieniku úsečky s touto priamkou
  2. určíme či v tomto prieniku úsečka vstupuje, alebo z okna vystupuje, t.j. priradíme prieniku príznak „*in*“ alebo „*out*“
  3. orezaná úsečka leží medzi posledným bodom, ktorým pôvodná úsečka vstupuje (*in*), a prvým, ktorým vystupuje (*out*), ak $t_{in} < t_{out}$
- Výhodou tohto postupu je, že koncové body orezanej úsečky počítame iba raz (na konci), pre ostatné body priebežne počítame len hodnotu parametra t
- Algoritmy využívajúce parametrické orezávanie je možné rozšíriť do 3D

Algoritmus Cyrus–Beck

- Ak platí $(B - A) \cdot \mathbf{n}_i < 0$, tak úsečka AB do okna vstupuje
- Ak platí $(B - A) \cdot \mathbf{n}_i > 0$, tak úsečka AB z okna vystupuje
- Priamka p ležiaca na niektorej strane okna má všeobecnú rovnicu: $(X - P) \cdot \mathbf{n}_p = 0$, kde P je nejaký bod na priamke (napr. niektorý roh okna)
- Bod $X = A + (B - A)t$ leží na priamke p, ak $((A + (B - A)t) - P) \cdot \mathbf{n}_p = 0 \Rightarrow t = \frac{(P-A)\cdot\mathbf{n}_p}{(B-A)\cdot\mathbf{n}_p}$
- Po výpočte všetkých prienikov nájdeme hodnoty $t_{in} = \max\{\{t_i \mid (B - A) \cdot \mathbf{n}_i < 0\}, 0\}$ a $t_{out} = \min\{\{t_i \mid (B - A) \cdot \mathbf{n}_i > 0\}, 1\}$
- Ak $t_{in} < t_{out}$, tak vypočítame súradnice koncových bodov $A + (B - A)t_{in}$ a $A + (B - A)t_{out}$ orezanej úsečky, inak úsečka leží mimo okna



Algoritmus Liang–Barsky

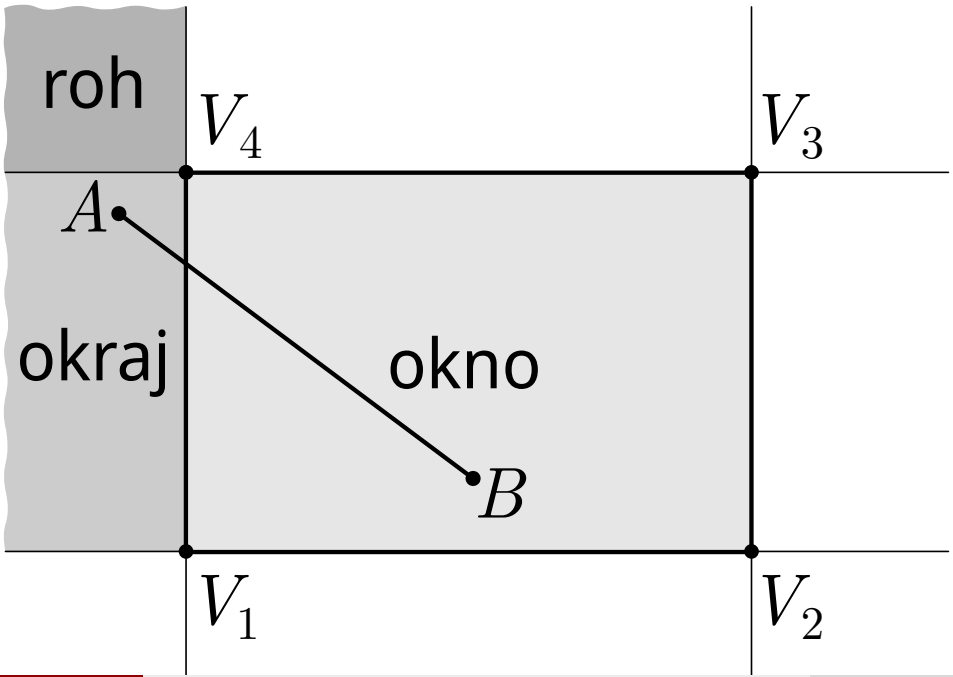
- Tento algoritmus je vylepšenie parametrického orezávania, ktorí využíva fakt, že strany okna sú rovnobežné so súradnicovými osami
- Bod $X = A + (B - A)t$ úsečky AB leží v okne práve vtedy, keď $x_{min} \leq a_x + \Delta x \cdot t \leq x_{max}$ a $y_{min} \leq a_y + \Delta y \cdot t \leq y_{max}$
- Upravme 4 predošlé nerovnice na tvar $p_i \cdot t \leq q_i$ pre $i = 1, 2, 3, 4$:

$$\begin{aligned} -\Delta x \cdot t &\leq a_x - x_{min} \\ \Delta x \cdot t &\leq x_{max} - a_x \\ -\Delta y \cdot t &\leq a_y - y_{min} \\ \Delta y \cdot t &\leq y_{max} - a_y \end{aligned}$$

- Ak $\Delta x = 0$ alebo $\Delta y = 0$, tak je úsečka rovnobežná s príslušnou stranou okna. Ak navyše aj $q_i < 0$, tak je celá úsečka mimo okna
- Ak pre niektoré i je $p_i < 0$, tak priamka príslušnou stranou vstupuje do okna
- Ak pre niektoré i je $p_i > 0$, tak priamka príslušnou stranou z okna vystupuje
- Ak $p_i \neq 0$, vypočítame parameter zodpovedajúci priesečníku: $t_i = q_i / p_i$
- Na záver určíme parametre pre orezanú úsečku $t_{in} = \max\{\{t_i \mid p_i < 0\}, 0\}$, $t_{out} = \min\{\{t_i \mid p_i > 0\}, 1\}$ a otestujeme či $t_{in} < t_{out}$

Algoritmus Nicholl–Lee–Nicholl

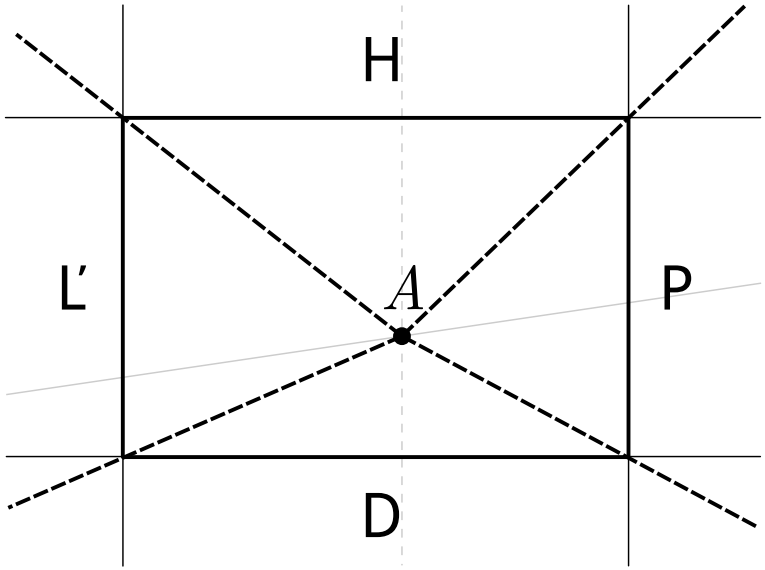
- V porovnaní s predošlými algoritmami počíta menej prienikov, avšak vykonáva veľa testov, aby sa vypočítali len tie priesečníky, ktoré sú naozaj potrebné
- Algoritmus **nie je** ľahko rozšíriteľný do trojrozmerného priestoru
- Predpokladá sa, že prvý bod úsečky sa nachádza v niektorom z troch regiónov – *roh*, *okraj*, *okno*
- Ak prvý bod padne do iného ako uvažovaného regiónu, je potrebné na okno aj úsečku aplikovať vhodnú transformáciu, rotáciu alebo zrkadlenie ($x' = \pm y$, $x' = -x$ a pod.) a po orezaní zasa inverznú transformáciu

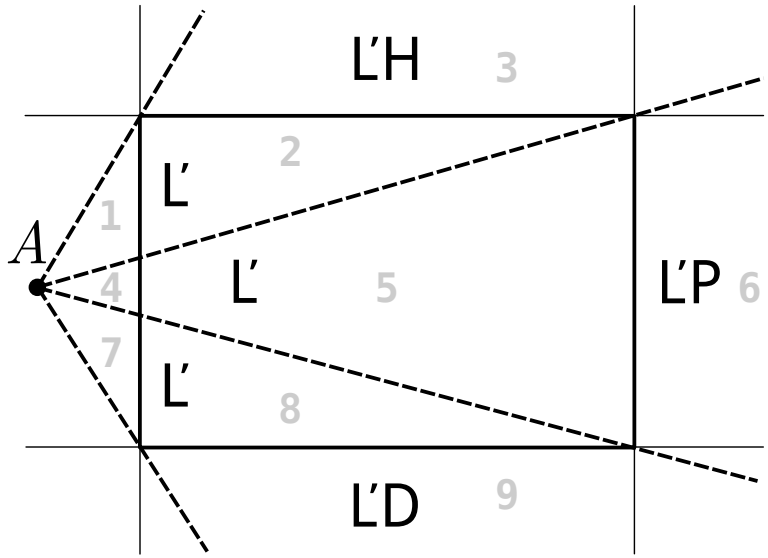


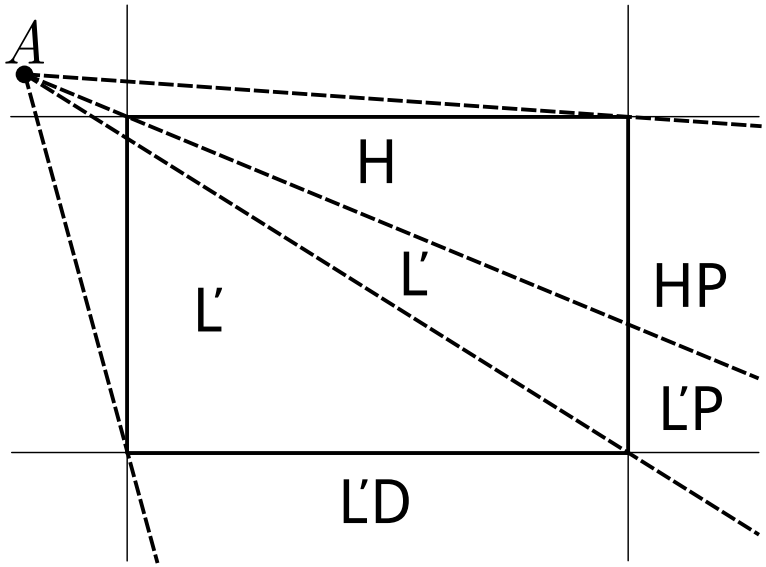
Algoritmus Nicholl–Lee–Nicholl

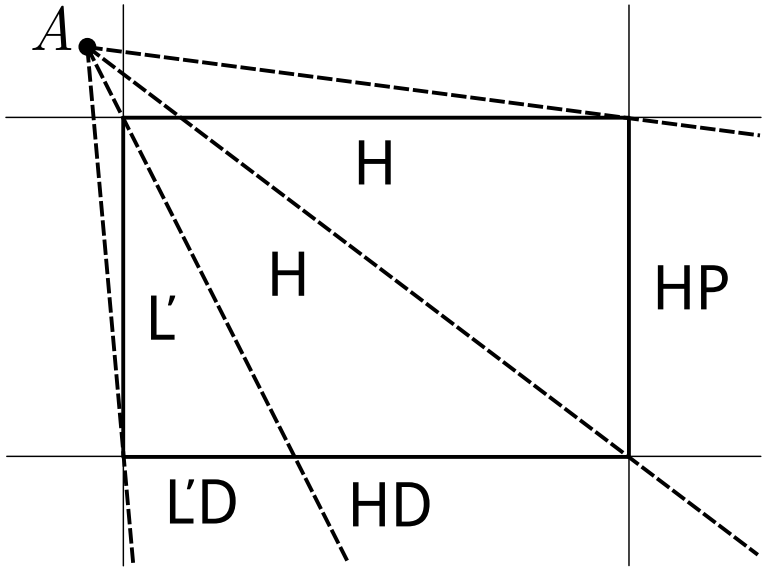
- Ak sa prvý bod úsečky nachádza v jednom z troch regiónov (*roh*, *okraj*, *okno*), existujú len 4 možné rozdelenia plochy
- Polpriamky a strany okna rozdelia plochu na 8 alebo 9 oblastí, podľa toho kde sa nachádza prvý bod úsečky
- Pri hľadaní, v ktorej oblasti leží orezávaná úsečka $AB : y = k \cdot x + q$, sa využíva jej sklon $k = \frac{b_y - a_y}{b_x - a_x}$
- Na obrázkoch sú oblasti označené jedným alebo dvoma písmenami, ktoré určujú akými stranami okna treba úsečku orezať (Ľ – ľavou, P – pravou, H – hornou, D – dolnou), keď sa druhý koncový bod nachádza v niektorej z oblastí

Algoritmus Nicholl–Lee–Nicholl









Algoritmus Nicholl–Lee–Nicholl

1. Určíme, v ktorom regióne (*roh*, *okraj*, *okno*) vzhľadom na okno sa nachádza prvý koncový bod úsečky
2. Rozdelíme celú 2D plochu na viacero oblastí. Použijeme strany okna a polpriamky, ktoré idú z prvého bodu úsečky postupne cez všetky rohy okna
3. Určíme, v ktorej oblasti leží priamka AB, porovnaním sklonu úsečky so sklonmi polpriamok, idúcim k rohovým bodom okna V_i a V_{i+1}:

$$\frac{V_{i,y} - a_y}{V_{i,x} - a_x} < \frac{b_y - a_y}{b_x - a_x} < \frac{V_{i+1,y} - a_y}{V_{i+1,x} - a_x}$$

   Ak je prvý bod v okne, musíme určiť, či je druhý od neho vpravo alebo vľavo, keďže sklon nám nedáva jednoznačnú informáciu a navyše sa pre orezanie H alebo D mení smernica z kladnej na zápornú
4. Po nájdení oblasti musíme porovnaním súradníc bodu B so súradnicami hranice okna zistiť, či úsečka pretína jednu alebo dve strany okna
5. Ak úsečka neleží v žiadnej oblasti, tak leží mimo okna
6. V inom prípade vypočítame 1 alebo 2 priesečníky z parametrického vyjadrenia úsečky a úsečku orežeme

Algoritmus Nicholl–Lee–Nicholl – príklad

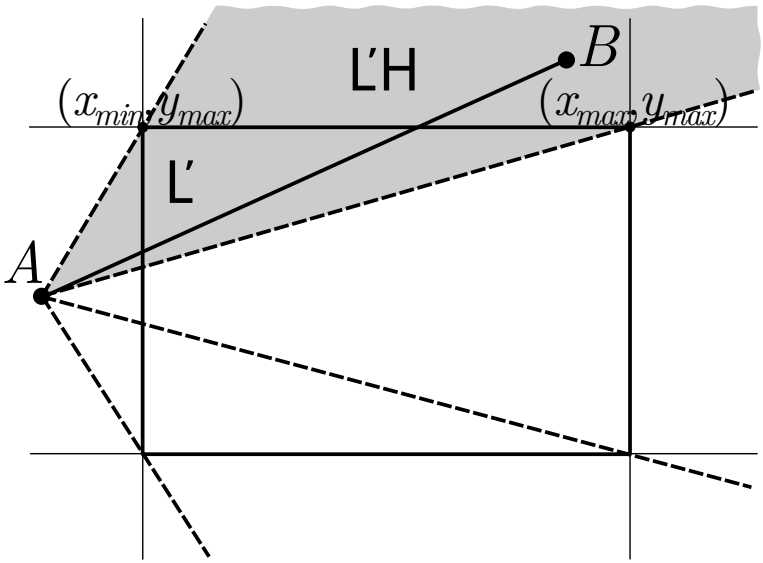
Na obrázku leží bod $B(b_x, b_y)$ v oblasti označenej ĽH, ak platí, že $\frac{y_{max}-a_y}{x_{max}-a_x} < \frac{b_y-a_y}{b_x-a_x} < \frac{y_{max}-a_y}{x_{min}-a_x}$ (Ľ alebo ĽH) a $b_y > y_{max}$ (ĽH)

Pri orezávaní ľavou stranou okna $x = x_{min}$ dostávame:

$$x_{min} = a_x + t(b_x - a_x) \Rightarrow t = \frac{x_{min} - a_x}{b_x - a_x} \Rightarrow x_p = x_{min},\ y_p = a_y + \frac{x_{min} - a_x}{b_x - a_x}(b_y - a_y)$$

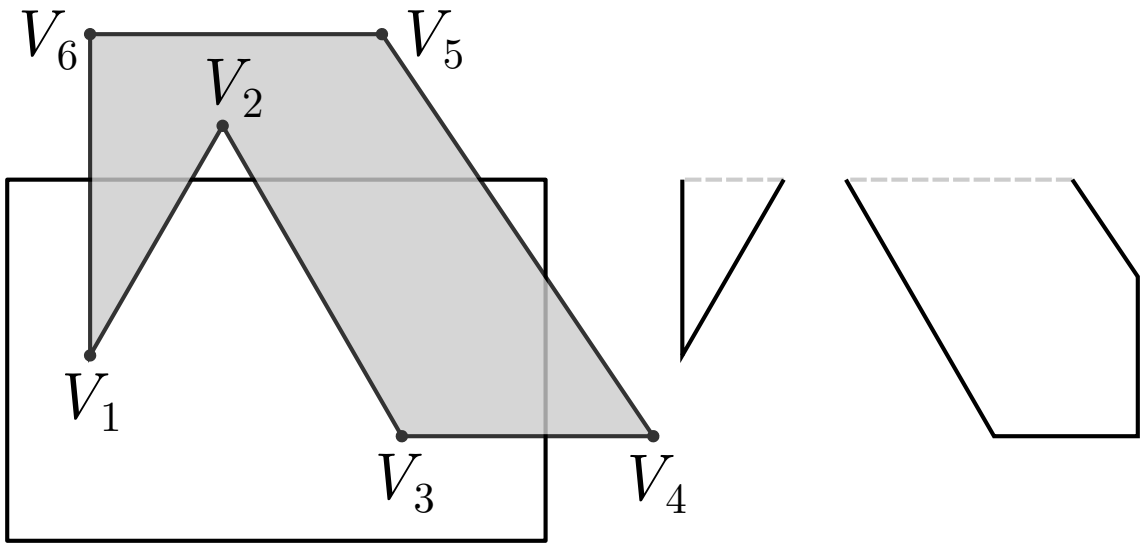
a pri orezávaní hornou stranou okna $y = y_{max}$:

$$y_{max} = a_y + t(b_y - a_y) \Rightarrow t = \frac{y_{max} - a_y}{b_y - a_y} \Rightarrow y_p = y_{max},\ x_p = a_x + \frac{y_{max} - a_y}{b_y - a_a}(b_x - a_x)$$



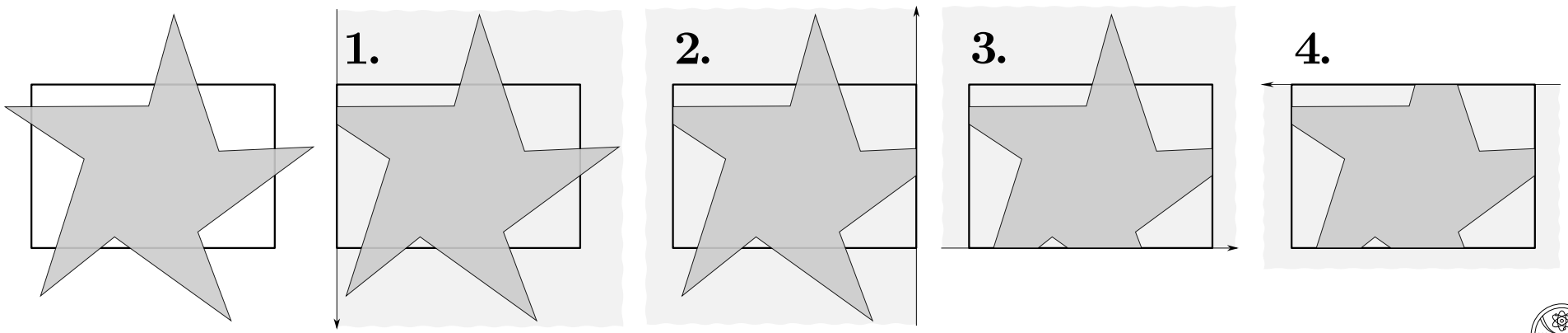
Orezávanie mnohouholníka

- Úlohou je pre mnohouholník zadaný zoznamom vrcholov nájsť prienik s oknom, pričom pri nekonvexnom mnohouholníku môže vzniknúť aj viac ako jeden orezaný mnohouholník
- Nestačí iba použiť metódu orezávania úsečiek na hrany mnohouholníka, pretože by vznikla postupnosť nespojených orezaných úsečiek.
- Orezané mnohouholníky musia byť po orezaní ohraničenou oblasťou, čo znamená, že musíme orezané úsečky pospájať
- Výstupom orezaného mnohouholníka by mala byť opäť postupnosť vrcholov



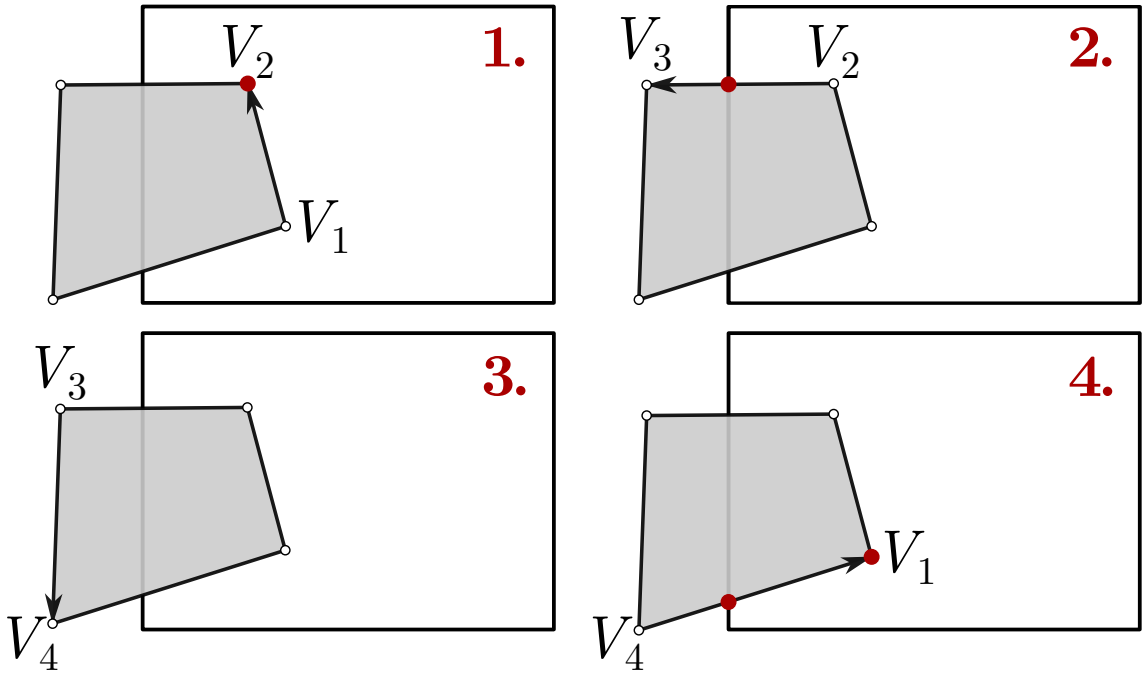
Algoritmus Sutherland–Hodgeman

- Využíva myšlienku, že konvexné okno je tvorené prienikom polpriesorov
- Orezáva hranice mnohouholníka ako celku, postupne s každou stranou okna
- Postupne sa spracovávajú všetky vrcholy práve orezaného mnohouholníka voči aktuálnej strane
- Začne sa ľavou stranou okna a oreže sa celý mnohouholník voči tejto strane
- Aktuálne orezaný mnohouholník sa ďalej oreže pravou, potom dolnou a nakoniec hornou stranou okna
- V každom kroku sa vygeneruje nová postupnosť vrcholov určujúca čiastočne orezaný mnohouholník, ktorý je potom orezávaný ďalšou stranou okna



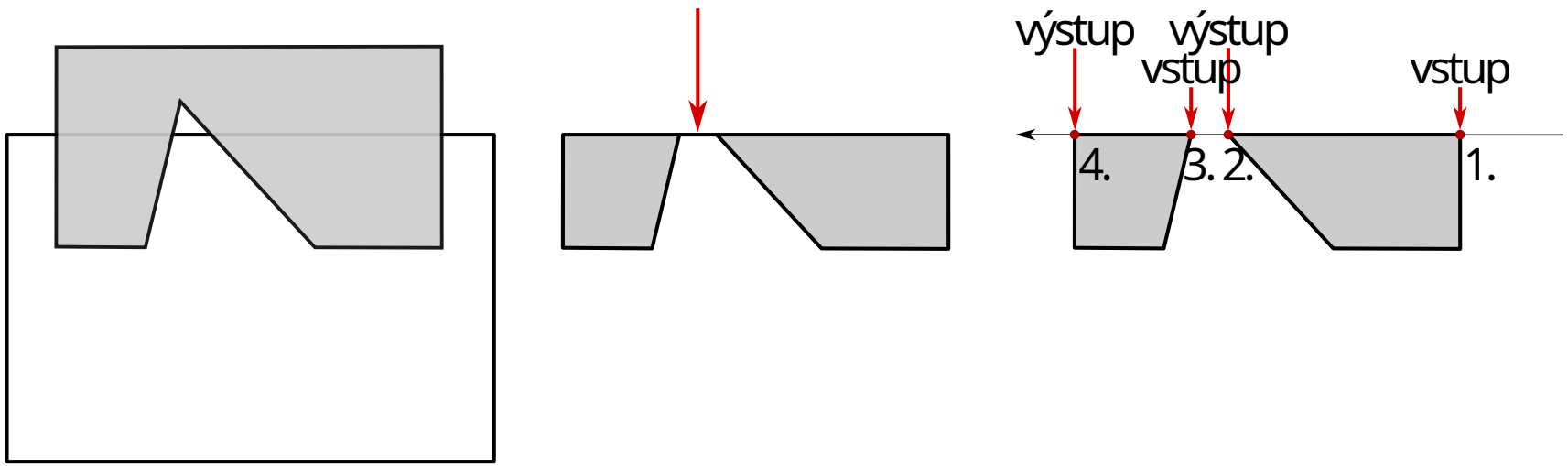
Algoritmus Sutherland–Hodgeman

- Pri prieniku hrany mnohouholníka so stranou okna existujú 4 možné prípady:
  1. ak sú oba vrcholy hrany vo vnútri aktuálnej orezávacej polroviny okna, pridáme do výstupného zoznamu vrcholov iba druhý vrchol,
  2. ak je prvý vrchol vo vnútri a druhý vonku, do zoznamu výstupných vrcholov sa pridá len priesečník hrany s hranicou okna,
  3. ak sú oba vrcholy mimo orezávacej polroviny, do zoznamu výstupných vrcholov sa nič nepridá,
  4. ak je prvý vrchol mimo a druhý vrchol je vo vnútri, do zoznamu výstupných vrcholov sa pridá priesečník s hranicou okna aj druhý vrchol



Algoritmus Sutherland–Hodgeman

- Algoritmus funguje aj pre ľubovoľné mnohouholníkové ale konvexné okno a dá sa upraviť aj pre 3D
- Správne orezáva konvexné mnohouholníky, ale nekonvexné môžu byť zobrazené s vonkajšími čiarami
- Keďže existuje iba jeden výstupný zoznam vrcholov, dva priesečníky sú spojené hranou aj vtedy, kedy by nemali byť spojené
- Jedným z možných riešení tohto problému je usporiadať priesečníky v smere orezávacej strany a použiť princíp parity vstup/výstup



Algoritmus Weiler–Atherton

- Môže sa použiť aj pre **nekonvexné** mnohouholníky, je rozšírením algoritmu Sutherland–Hodgeman
- Základnou myšlienkou tohto algoritmu je, že namiesto toho, aby sme pri spracovávaní vrcholov postupovali okolo hrán mnohouholníka, budeme postupovať **aj** po hranici okna
- Použité pravidlá v algoritme závisia od:
  - zadaného sledu vrcholov mnohouholníka (v smere hodinových ručičiek alebo proti smeru),
  - orientácie dvojice vrcholov mnohouholníka ku oknu (zvonka-dovnútra alebo zvnútra-vonku)
- Ak sú vrcholy mnohouholníka zadané v smere hodinových ručičiek používame nasledujúce pravidlá:
  1. pre dvojicu vrcholov, ktorá ide zvonka dovnútra, sa prejde na hranicu mnohouholníka a pokračuje sa v smere hodinových ručičiek,
  2. pre dvojicu vrcholov, ktorá ide z vnútra von, sa prejde na hranicu okna a pokračuje sa v smere hodinových ručičiek
- Keďže tento algoritmus orezávania mnohouholníkov nefunguje pre samopretínajúce sa mnohouholníky, boli navrhnuté metódy, ktoré riešia aj tento problém

Algoritmus Weiler–Atherton pre prienik dvoch polygónov

1. Najprv vytvoríme zoznam všetkých priesečníkov P_1, P_2, P_3, ...
2. Ak je zoznam priesečníkov prázdny, treba overiť vzájomnú polohu mnohouholníkov, či jeden mnohouholník neleží v druhom. Stačí otestovať či jeden vrchol mnohouholníka neleží v druhom a opačne
3. Oklasifikujeme každý priesečník ako vstupný alebo výstupný. Ak je priesečník pre prvý mnohouholník vstupný, potom je pre druhý mnohouholník výstupný a naopak
4. Vytvoríme dva zoznamy, jeden pre **orezávajúci** (ktorým orezávame) mnohouholník a druhý pre **orezávaný** (ktorý budeme orezávať) mnohouholník
5. Doplníme oba zoznamy o priesečníky tak, aby priesečníky ležali medzi správnymi vrcholmi oboch mnohouholníkov
6. Začneme zoznamom vrcholov orezávaného mnohouholníka a vyberieme prvý nespracovaný priesečník, ktorý bol označený ako vstupný a zároveň ho označíme ako spracovaný
7. Ďalej postupujeme ďalšími vrcholmi v zozname (v prípade, že sme na konci zoznamu, prejdeme na prvý vrchol v zozname) orezávaného mnohouholníka, až kým sa dostaneme priesečník, ktorý je označený ako výstupný

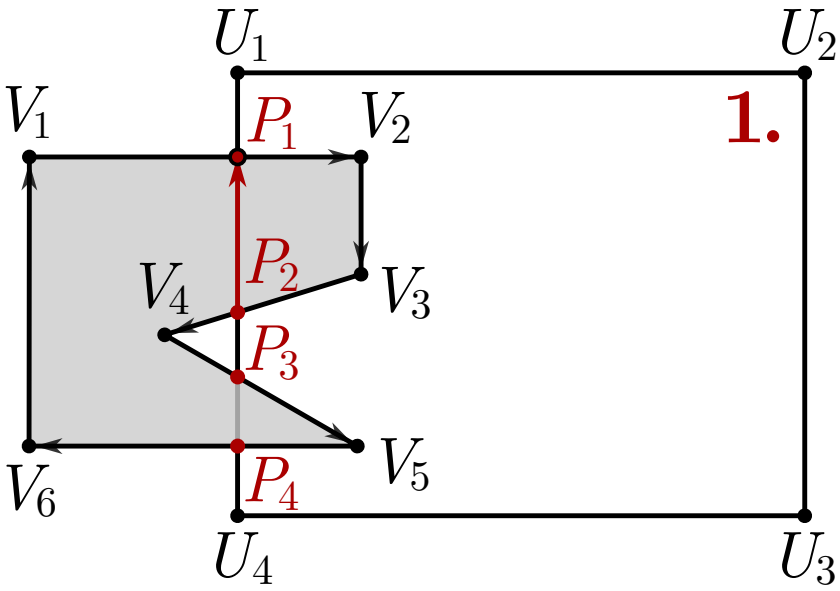
Algoritmus Weiler–Atherton pre prienik dvoch polygónov

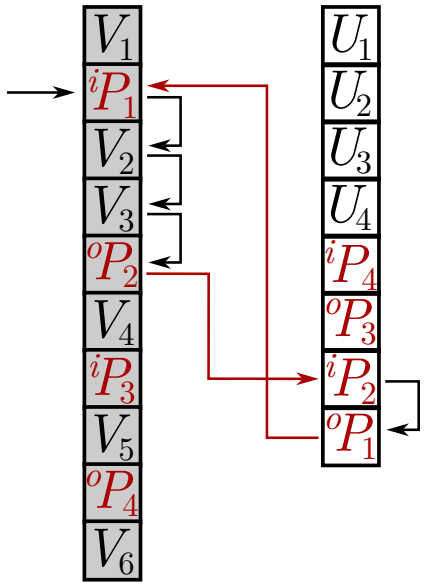
7. Označíme tento priesečník ako spracovaný a prejdeme na zoznam vrcholov orezávajúceho mnohouholníka a nájdeme v jeho zozname aktuálne spracovávaný priesečník
8. Pokračujeme vrcholmi v zozname orezávajúceho mnohouholníka, až kým nenájdeme priesečník označený ako výstupný
9. Označíme tento priesečník ako spracovaný a prejdeme na zoznam vrcholov orezávaného mnohouholníka
10. Takto prechádzame z jedného mnohouholníka/zoznamu na druhý (t.j. opakujeme kroky 6 až 9), až kým sa dostaneme k už spracovanému priesečníku. Vtedy dostaneme výstupný orezaný polygón, pričom ak je orezávaný mnohouholník nekonvexný, tak môže byť jeden z viacerých
11. Opakujeme tento postup orezávania od kroku 5, až kým navštívime všetky nespracované vstupné priesečníky

Algoritmus Weiler–Atherton - príklad

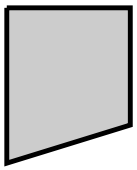
1. Vytvoríme zoznam vrcholov pre orezávajúci a orezávaný mnohouholník, vypočítame priesečníky P_1, P_2, P_3, P_4, oklasifikujeme ich a vložíme ich na správne miesto zoznamu vrcholov oboch mnohouholníkov
2. Začneme od vstupného priesečníka P_1, zapíšeme ho zoznamu výstupných vrcholov, označíme ho ako spracovaný a posunieme sa ku vrcholu V_2, ktorý tiež zapíšeme do výstupného zoznamu vrcholov
3. Pri prechode od vrchola V_2 ku V_3 zapíšeme do výstupného zoznamu vrchol V_3
4. Pokračujeme v zozname vrcholov orezávaného mnohouholníka ku výstupnému priesečníku P_2, ktorý označíme ako spracovaný, zapíšeme ho do nového zoznamu a prejdeme na zoznam vrcholov okna
5. Ďalej pokračujeme vrcholmi okna až sa dostaneme do už označeného priesečníka P_1, čím získame prvú orezanú časť
6. V zozname vrcholov orezávaného mnohouholníka nájdeme ďalší nespracovaný vstupný priesečník P_3, označíme ho ako spracovaný a pridáme ho do výstupného zoznamu spolu s nasledujúcim vrcholom V_5
7. Ďalej sa presunieme do výstupného priesečníku P_4, pridáme ho do výstupného zoznamu a prejdeme na hranicu okna. Pokračujeme priesečníkom P_3, ktorý je už ale označený ako spracovaný, preto získame druhú orezanú časť

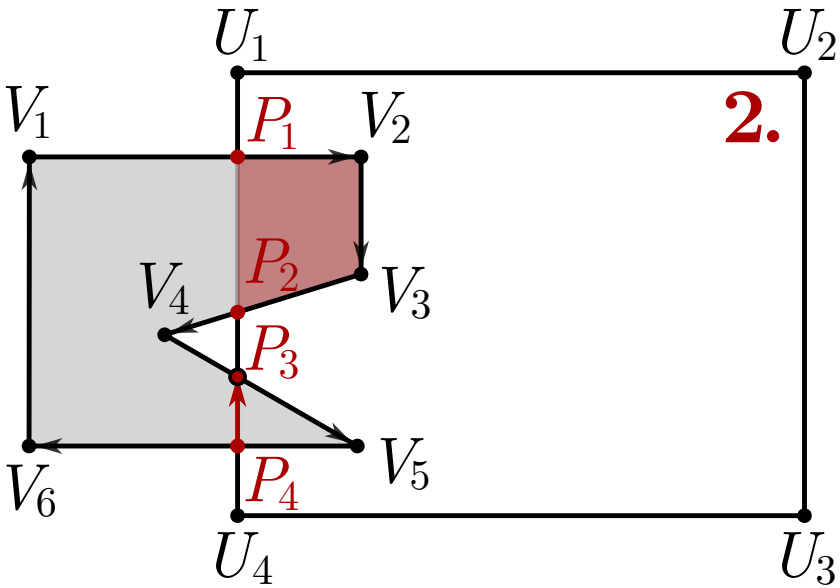
Algoritmus Weiler–Atherton - príklad

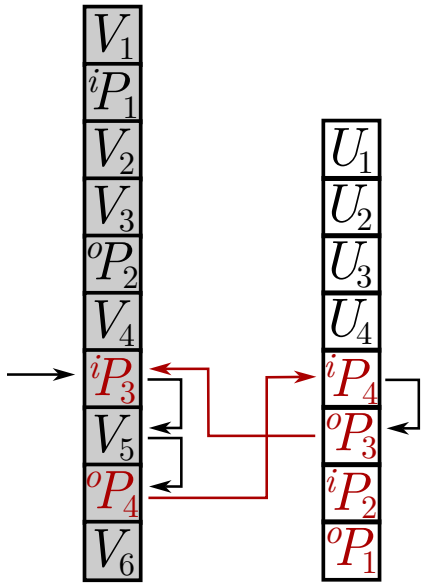




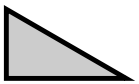
$P_1\ V_2\ V_3\ P_2\ P_1$





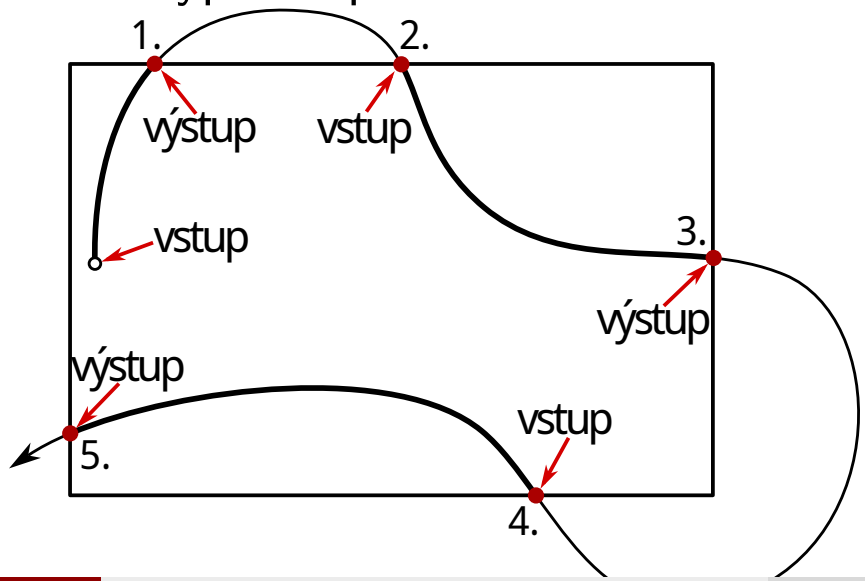


$P_3\ V_5\ P_4\ P_3$



Orezávanie krivky

- Vhodné je najskôr namiesto orezávania overiť, či má krivka s oknom prienik. Môže sa použiť obálka vo forme obdĺžnika alebo konvexného mnohouholníka a potom sa testuje vzťah mnohouholníka a okna
- Ak je krivka určená polynómami, môže sa pre výpočet priesečníku použiť Beziérovo orezávanie
- Keď sú známe priesečníky, treba určiť, ktoré časti krivky sú vo vnútri okna. Ak usporiadame body prieniku pozdĺž smeru krivky a určíme polohu prvého bodu krivky, môžeme opäť použiť princíp parity vstup/výstup
- Ak je krivka vykresľovaná ako sled úsečiek, potom sa výpočet bodu prieniku krivky a okna zmení na výpočet prieniku úsečiek a okna



Orezávanie krivky, mnohouholníka nekonvexným oknom

- Ak nie je možné použiť komplikovanejšie algoritmy, ktoré pracujú aj s nekonvexným oknom, doporučuje sa rozdeliť nekonvexné okno na konvexné časti
- Po vypočítaní prieniku krivky so všetkými konvexnými časťami, je nutné spojiť orezané časti krivky dokopy, napr. podľa susednosti vzniknutých konvexných častí