

REFERENCES

1. A. K. Raina, 'Importance and sensitivity of variables defining the performance of pre-split blasting using artificial neural networks', *Mining Metallurgy Exploration*, vol. 38, no. 4, pp. 1817–1829, 2021.
2. A. P. Krasavin, Y. M. Vasil'kov, A. Y. Davankov, and A. N. Shaufler, 'Land reclamation using draglines. Rekul'tivatsiya Narushennykh Zemel's Ispol'zovaniem Draglainov', 1983.
3. A. R. Sahu and S. K. Palei, 'Fault prediction of drag system using artificial neural network for prevention of dragline failure', *Eng Fail Anal*, vol. 113, p. 104542, 2020.
4. B. A. Kennedy and B. A. Kennedy, *Surface mining*. SME, 1990.
5. B. T. Jones, 'Dragline maintenance engineering', 2007.
6. Baafi, E.Y.; Mirabediny, H. and Whitchurch, K. (1997). Computer simulation of complex dragline operations, *International Journal of surface Mining, Reclamation and Environment*, Vol.11, No.1 p.7-13.
7. Bandopadhyay, S. and Ramani, R.V. (1979). Computer simulation of dragline development schemes, *Proc. 16th Int. Symp. Application Comp. & Op. Res. Min. Ind.*, Tucson, AR.
8. Bandopadhyay, S. and Ramani, R.V. (1985). Simulation of a dragline operation in an eastern Kentucky mine, *C.I.M. Bull.*, Vol.78, No.10, Oct., p.52-58.
9. Carter, R.A. (1989). Software shows graphic improvement, *Coal*, July, Vol.26, No.7, p. 50-53.
10. Central Mine Planning & Design Institute Limited (2021), Ranchi, India.
<https://www.cmpdi.co.in/en>

11. Chaoji, S.V. and Dey, B.C. (2000). Dragline operation in mines - an overview, *Jl. of Mines, Metals & Fuels*, Vol. XLVIII, No. 5, p. 84-93.
12. Chatterjee, P.K., Rowland, D. and Siller, K.C. (1977). Simulation of dragline operations, *Proc. 14th Int. Symp. on the Application of Comp. Methods in the Mineral Industry*, Ramani, R.V., ed., Soc. of Mining Engineers, SME/AIME, p.584-594.
13. Chironis, N.P. (1986). Smaller, Versatile Dragline Operating in Tandem, Reach Deep Seams Efficiently, *Coal Age*, December, p. 48-51.
14. Chugh, Y.P. (1980). Surface mining of minerals by dragline in the U.S.A., *Trans. Instn Min. Metall. Sec. A*, Vol. 89, October, p. A198-A204.
15. Coal directory of India (2021). Coal Statistics, Government of India, Ministry of Coal, Coal Controller's Organization, Kolkata.
16. Coal India Limited (2021), Kolkata, India. <https://www.coalindia.in/>
17. Corke, P., Winstanley, G., Dunbabin, M., Roberts, J.(2006). Dragline automation: Experimental evaluation through productivity trial. *Springer Tracts Adv. Robot.* 24, 459– 468.
18. D. Measles and N. Bugosh, 'Making and building a fluvial geomorphic reclamation design at an active dragline mine using the Geo-FluvTM design method', *30 Years of SMCRA and Beyond*, pp. 2–7, 2007.
19. Dragline Dictionary, 4th Edition, 2014, PwC – Mining Intelligence and Benchmarking
20. E. Y. Baafi, H. Mirabediny, and K. Whitchurch, 'A simulation model for selecting suitable digging method for a dragline operation', 1995.
21. Erdem, B. and Celebi, N. (1999). A method to synchronize the tandem dragline systems, *CIM Bulletin*, Vol. 92, No. 1029, p. 55-61.

22. Erdem, B., Celebi, N. and Pasamehmetoglu, A.G. (1998). Optimum Dragline Selection for Strip Coal Mines, Trans. Instn. Min. Metall. Sect. A, Vol. 107, January-April, p. A13-A24.
23. F. v Dudinskiy and K. B. Nechaev, 'Land reclamation technologies for direct dumping', in IOP Conference Series: Earth and Environmental Science, 2020, vol. 408, no. 1, p. 012051.
24. Fishler, S.V. (1987). Selection of the most cost effective dragline system, International Journal of Surface Mining, Vol.1, p.91-95.
25. G. Walton and T. Atkinson, 'Some geotechnical considerations in the planning of surface coal mines', Inst. Min. Metall., Trans., Sect. A;(United Kingdom), vol. 88, 1979.
26. H. George, J. Meech, and L. Workman, 'Towards reducing the physical environmental impact of North American surface coal mines; a review of potential selective overburden handling techniques', Mining Science and Technology, vol. 3, no. 2, pp. 81–94, 1986.
27. H. Mirabediny, 'A dragline simulation model for strip mine design and development', 1998.
28. Hrebar, M. J., and K. Dagdelen. "Equipment selection using simulation of dragline stripping methods." 16 th APCOM Symposium, Soc. Min. Eng., AIME. 1979.
29. Hrebar, M.J. and Cook, H.C. (1987). Estimating dragline productivity using a graphic microcomputer program, International Journal of Surface Mining, Vol.1, p.251-255.
30. Huddart, H.C. and Runge, I.C. (1979). Pit design optimised for dragline productivity, Inst. of Min. and Met. Sec. A., Vol. 88, 1979, p.A-6-A-12.

31. Humphrey, J.D. 1990. The Fundamentals of the Dragline, Marion Division-Dresser Ind. Inc. Marion, Ohio, USA, pp. 1-28.
32. Industrial dragline Source : <https://mining.komatsu/en-in/product-details/p-h-9020c>
33. J. D. Humphrey, 'The fundamentals of the dragline', Marion Power Shovel Division Dresser Industries: Marion, OH, USA, 1990.
34. J. N. Adams, 'UK opencast mining equipment in the 1980's', International Journal of Surface Mining, Reclamation and Environment, vol. 4, no. 3, pp. 115–118, 1990.
35. Jameson, I.J. and Thomas, M.D. (1992). Selection of a P& H 757 dragline, Min. Technology, Vol. 74, No. 852, Feb., p. 36-40.
36. Johnson, T.C. (1974). How design improvements boost walking dragline productivity, Soc. of Min. Engineers, Oct., p. 46-49.
37. Jones, T., Blackmore, P., Leach, M., Bérubé, K., Sexton, K., Richards, R. (2002). Characterisation of airborne particles collected within and proximal to an opencast coalmine: South Wales, U.K. Environ. Monit. Assess. 75, 293–312.
38. Kishore, N., & Jaiswal, A. (2004). Design And Planning Of Dragline Pits A Few Consideration. Mining Engineers Journal, 5(10), 26-29.
39. Kovalev, V. A. and Fedotenko, V.S. Technological Aspects of Transition to High Bench Stripping in Kuzbass, Journal of Mining Science, 2015, Vol. 51, No. 5, pp 865 – 872. V.I.
40. L. H. Michaud, 'Applications of interactive computer graphics for dragline mine planning in mountainous terrain.', 1993.

41. Lal, M., Agrawal, M.R., and Rai, R.L. (1988). Suggested method of working for Jayant opencast mine operation of dragline in tandem, National Symp. on Mine Planning and Design, B.H.U., Varanasi, Nov., p.1.01-1.10.
42. Learmont, T. (1975). Productivity Improvement in Large Stripping Machines, Transactions SME, Vol. 258.
43. Learmont, T. (1989). The Walking Dragline and its Application
44. Lee, C.D. (1988). A multiseam dragline mine planning simulator for personal computers, Int. Symp. on Mine Planning and Equipment Selection, Singhal (Ed.), Balkema, Calgary, Canada, p.397-408.
45. M. A. Wardeh and S. Frimpong, 'Kinematic analysis of an under-actuated, closed-loop front-end assembly of a dragline manipulator', International Journal of Automation and Computing, vol. 17, no. 4, pp. 527–538, 2020.
46. M. J. Bertoldi, Preliminary Economics of Mining a Thick Coal Seam by Dragline, Shovel-truck, and Scraper Mining Systems, vol. 8761. Department of the Interior, Bureau of Mines, 1977.
47. M. J. Hrebar, 'Preliminary dragline selection for surface coal mining operations, Mine Planning and Equipment Selection', Singhal and Vavra (eds.) Balkema, Rotterdam, 1990.
48. M. P. Clarke, B. Denby, and D. Schofield, 'Decision making tools for surface mine equipment selection', Mining Science and Technology, vol. 10, no. 3, pp. 323–335, 1990.
49. McLean, P. and Baldwin, G. (1989), Dragline Monitoring at German Creek Mine, Draglines - Operation, Maintenance and Control, ACIRL Proceedings, Monograph QLD, pp: 139-150.

50. Michaud, L.H. and Calder, P.N. (1993). The development of a computerized dragline mine planning package utilizing interactive computer graphics, CIM Bulletin, Vol. 86, No. 973, p. 37-42.
51. Ministry of Power, Government of India, Central Electricity Authority (CEA), 2021, New Delhi.
52. Mohammadi, Mousa, and Piysh Rai. "Improving Performance of Mining Equipment Through Enhancement of Speed Factor: A Case Study (Research Note)." International Journal of Engineering 28.9 (2015): 1392-1400.
53. Morey, P.G. and Lee, C.D. (1979). Optimization of Dragline Operation, Mining Congress Jl., p. 27-30.
54. N. Demirel, 'Dynamic dragline modelling and boom stress analysis for efficient excavation', 2007.
55. Northern Coalfields Limited (2021), Singrauli, India. <http://nclcil.in/>
56. O. Patir and H. Salgir, 'Cast blasting in coal mine dragline applications; Dragline dilimlerinde patlatma ile malzeme otelenmesi', 1996.
57. P. Darling, SME mining engineering handbook, vol. 1. SME, 2011.
58. P. Harder, 'Dozers, draglines, and cast blasting', Coal;(United States), vol. 25, no. 11, 1988. (Harder,1988)
59. P. K. Chatterjee, D. Rowland, and K. C. Siller, 'Simulation of dragline operations', 1977.
60. Phillips, M. (1989), Utilisation of Data from Dragline Monitoring Systems, Draglines - Operation, Maintenance and Control, ACIRL Proceedings, Monograph QLD, pp: 106- 118
61. Provisional Coal Statistics (2021). Coal Controller's Organization, Ministry of Coal, Government of India, Kolkata.

62. Pundari, N.B. (1981). Selecting and Using Large Walking Dragline for Deeper Overburden Stripping, Mining Engineering, April, p. 377-381.
63. R. v Ramani, 'Surface mining technology: progress and prospects', Procedia Eng, vol. 46, pp. 9–21, 2012.
64. Rai P, Kishore N. and Nath R. (2001): Increasing Dragline Output-Cut Width Analysis in a Tandem Dragline Operation, World Mining Equipment. Vol.25, No.3, pp. 48-52
65. Rai, P. (2000). A critical analysis of draglines operating in horizontal tandem in opencast coal mines - a case study, The Indian Mining & Engineering Jl., Vol.39,p. 25-28.
66. Rai, P., Kishore, N. and Nath, P. (1999). Effects of cut width on performance of dragline operating in horizontal tandem - a case study, Coal Int. Mining & Quarry World, Vol. 247, No. 6, Oct./Nov., p. 216-220.
67. Rai, P., Kishore, N. and Nath, P. (2000). Influence of overburden thickness and width of key cut on the performance of draglines in horizontal tandem, Jl. of the Inst. of Engineers (India), Vol. 81, August, p. 10-13.
68. Rai, Piyush, Umakant Yadav, and Ashok Kumar. "Productivity analysis of draglines operating in horizontal and vertical tandem mode of operation in a coal mine—a case study." Geotechnical and Geological Engineering 29.4 (2011): 493-504.
69. Rai, R.L. (1990). An Approach to Tandem Operation (In vertical) of draglines with Due considerations for safety and conservation, Mine Tech., Vol.11, No.6, p. 7-10.
70. Rai, Sheo Shankar, Bhola, Singh (2021). A Survey of Dragline Mining Methods and Scopes for Productivity Improvement of Dragline Operation in India

(International Conference on Opencast Mining Technology & Sustainability, ICOMS-2021.

71. Rai, V.K. (1997). Balancing Diagram : A Basic planning Tool for Dragline Operation, Proceeding of the National Seminar “Emerging technology in surface Mines and Environmental challenges”, Manglore, p.85-89.
72. Rowlands, J.C., 1991. Dragline Bucket Filling, PhD thesis, University of Queensland.
73. Rzhnevsky, V.V. (1985). Opencast Mining Unit Operations, Mir Publishers, Moscow.
74. Rzhnevsky, V.V. (1987). “Opencast Mining Technology and Integrated Mechanization”, Mir Publishers, Moscow.
75. S. A. Tagieddin, ‘Applicability of blast casting technique in strip-mining phosphate rock’, Eng Geol, vol. 33, no. 2, pp. 127–139, 1992.
76. S. Akhundov and N. Demirel, ‘Development of a Computer-Aided Dragline Selection Program’, in Proceedings of the 27th International Symposium on Mine Planning and Equipment Selection-MPES 2018, 2019, pp. 235–244.
77. S. Bandopadhyay and R. v Ramani, ‘Digital simulation of dragline deployment schemes’, in Proc Sixteenth Int Symp Application Computer Methods Mineral Industries, 1979, pp. 431–448.
78. S. K. Ray, A. Zutshi, and M. Sarkar, ‘Cast blasting-state of the art’, Fragblast, vol. 3, no. 4, pp. 291–302, 1999.
79. S. S. Rai, V. Murthy, R. Kumar, M. Maniteja, and A. K. Singh, ‘Using machine learning algorithms to predict cast blasting performance in surface mining’, Mining Technology, pp. 1–19, 2022.

80. Sadri, R.J. (1982). Optimization of single and multiple seam dragline mines through simulation, proc., 17th. Int. Symp. on Applied of Computers and Operation Research in Mineral Industry.
81. SEERVI, VIKRAM, NAWAL KISHORE, and AMIT KUMAR VERMA. "Selection of mode of tandem dragline operations by utilizing 3-dimensional computer graphics balancing diagram: a case study."
82. Seervi, Vikram, Nilesch Pratap Singh, Nawal Kishore, and Amit Kumar Verma. "Critical Analysis of Tandem Dragline Performance in Open Cast Coal Mines-A Case Study." Journal of The Institution of Engineers (India): Series D (2022): 1-12.
83. Shevyakov, L. (1965). Mining and Mineral Deposits, Moscow Foreign Language Pub. House.
84. Singh, J.P., & Rai, V.K. (1998). Mine Planning for Maximising dragline Operation in Multi-Seam Stripping, Int. Conference by MGMI, Opencast "98", 331-338.
85. Stuart, N.J. and Cobb, Q. (1988). Two approaches to the computerized planning of dragline operations, Int. Symp. on Computer Applications in the Mineral Industry, Balkema, p.23-31.
86. T. Atkinson, 'Selection and sizing of excavating equipment', SME mining engineering handbook, vol. 2, pp. 1311–1333, 1992.
87. Tatiya, Ratan Raj. Surface and underground excavations: methods, techniques and equipment. CRC Press, 2005.
88. W. A. Crosby and K. L. McDonald, 'Productivity improvements for dragline operations using controlled blasting in a single-and multiple-seam opencast coal

- operation at Rietspruit, South Africa', CIM Bull. ;(Canada), vol. 76, no. 851, 1983.
89. W. Swenson et al., 'Managing Wisconsin fishponds', University of Wisconsin-Madison Publication G, vol. 3693, 2000.
90. Zhang, W., Cai, Q., Chen, S. (2013). Optimization of transport passage with Dragline system in thick OB surface mine. Int. J. Min. Sci. Technol. 23, 901–906.

APPENDIX-1 - THREE-DIMENSIONAL BALANCING DIAGRAMS

6.1 Annual production of leading dragline is less than the annual production of lagging dragline.

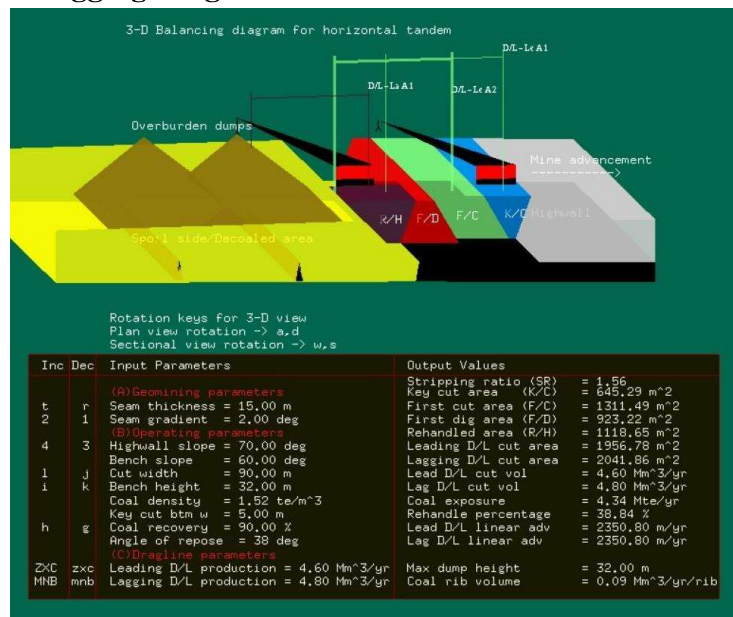


Figure 6.1.2: 3-dimensional balancing diagram for 4.60 Mm³ leading and 4.80 Mm³ lagging dragline.

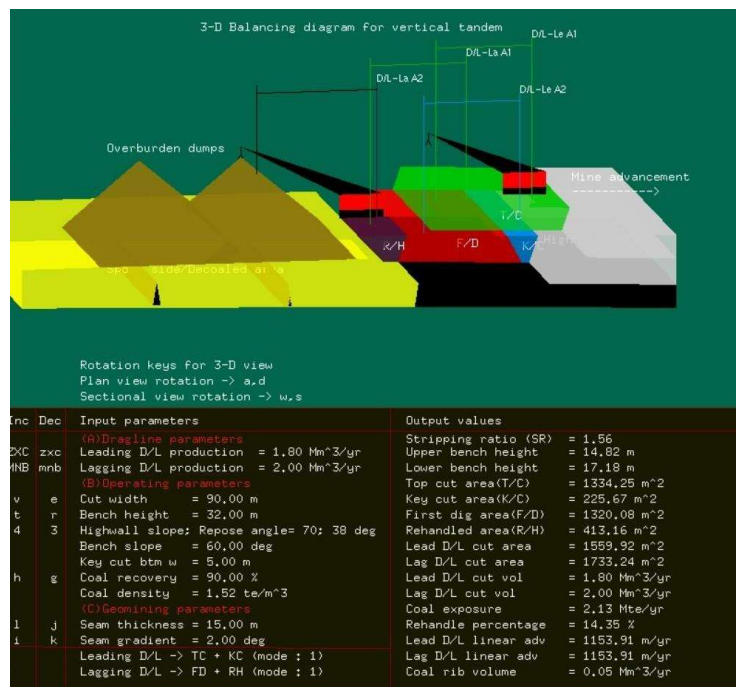


Figure 6.1.3: 3-dimensional balancing diagram for 1.80 Mm³ leading and 2.00 Mm³ lagging dragline.

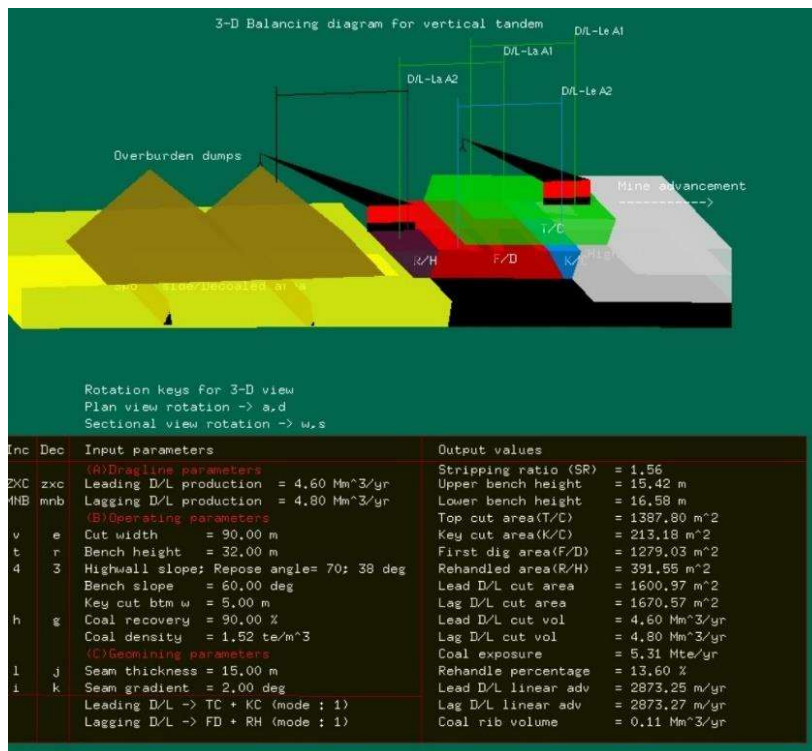


Figure 6.1.4: 3-dimensional balancing diagram for 4.60 Mm³ leading and 4.80 Mm³ lagging dragline.

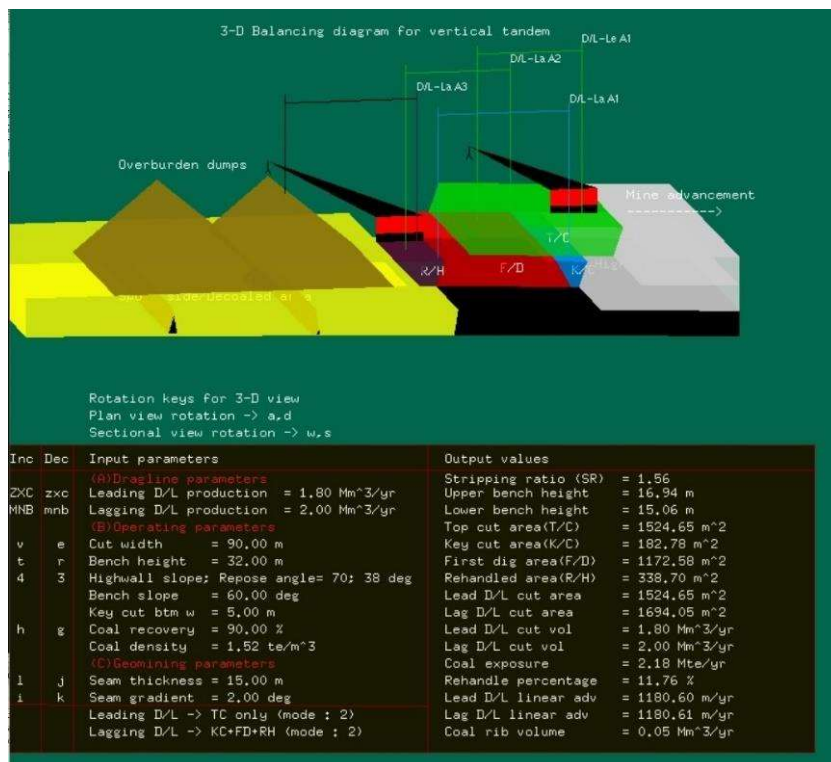


Figure 6.1.5: 3-dimensional balancing diagram for 1.80 Mm³ leading and 2.00 Mm³ lagging dragline.

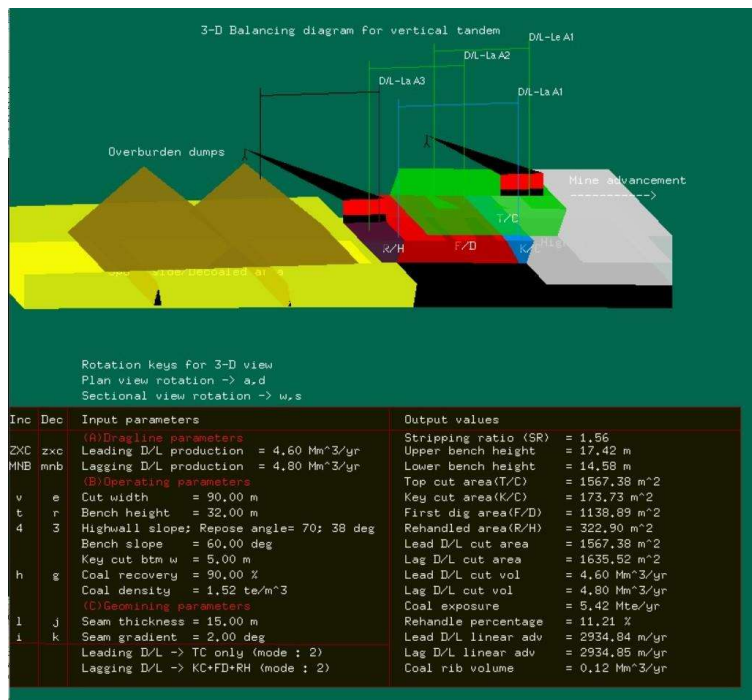


Figure 6.1.6: 3-dimensional balancing diagram for 4.60 Mm³ leading and 4.80 Mm³ lagging dragline.

6.2 Annual production capacity of leading and lagging dragline are same

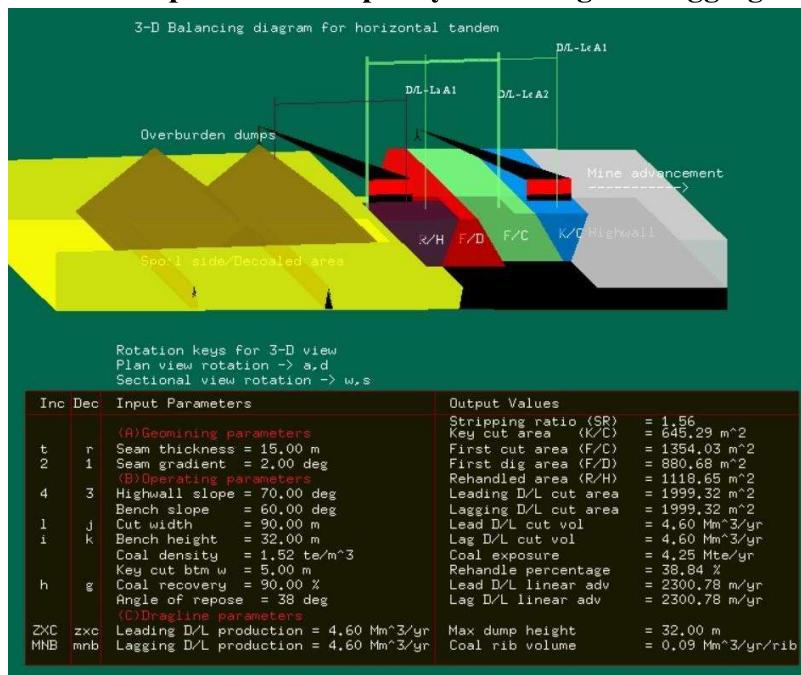


Figure 6.2.2: 3-dimensional balancing diagram for 4.60 Mm³ leading and lagging dragline.

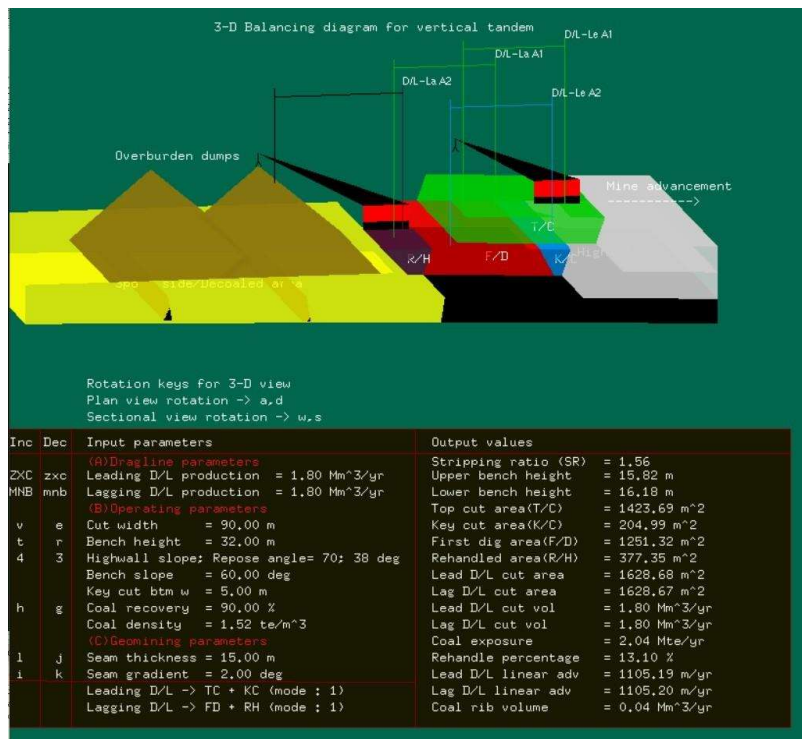


Figure 6.2.3: 3-dimensional balancing diagram for 1.80 Mm³ leading and lagging dragline.

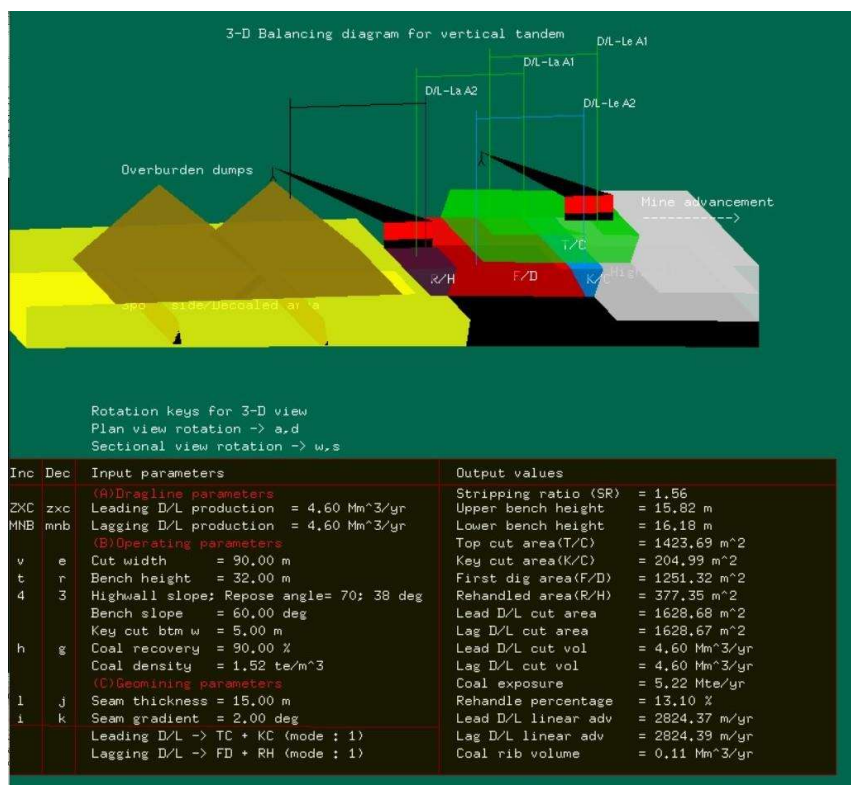


Figure 6.2.4: 3-dimensional balancing diagram for 4.60 Mm³ leading and lagging dragline.

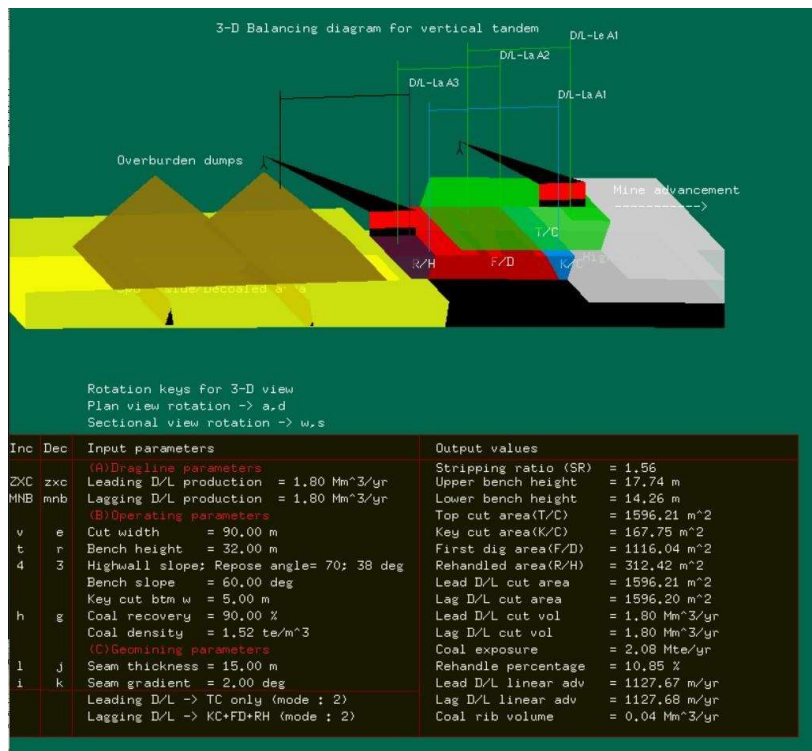


Figure 6.2.5: 3-dimensional balancing diagram for 1.80 Mm³ leading and lagging dragline.

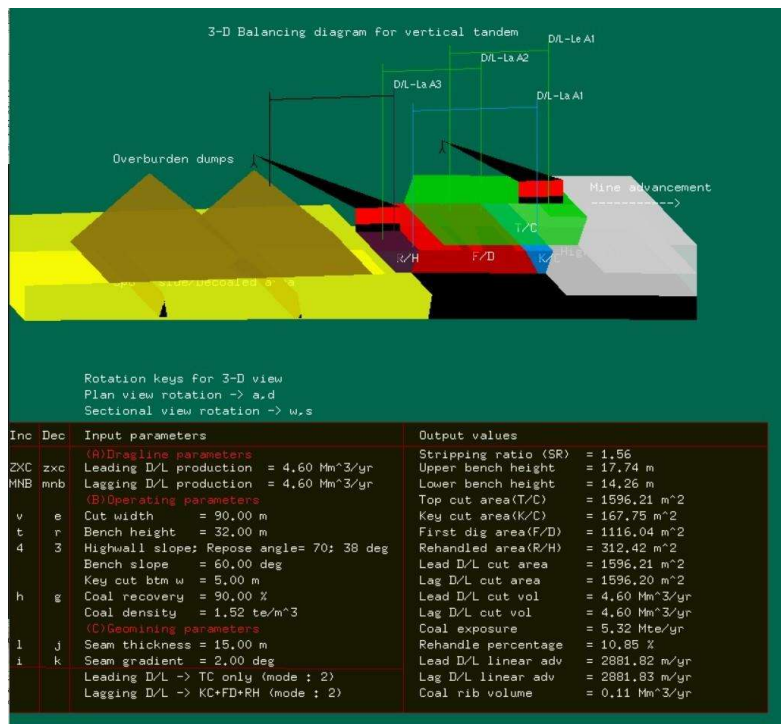


Figure 6.2.6: 3-dimensional balancing diagram for 4.60 Mm³ leading and lagging dragline.

6.3 Annual production capacity of leading dragline is greater than the annual production capacity of lagging dragline.

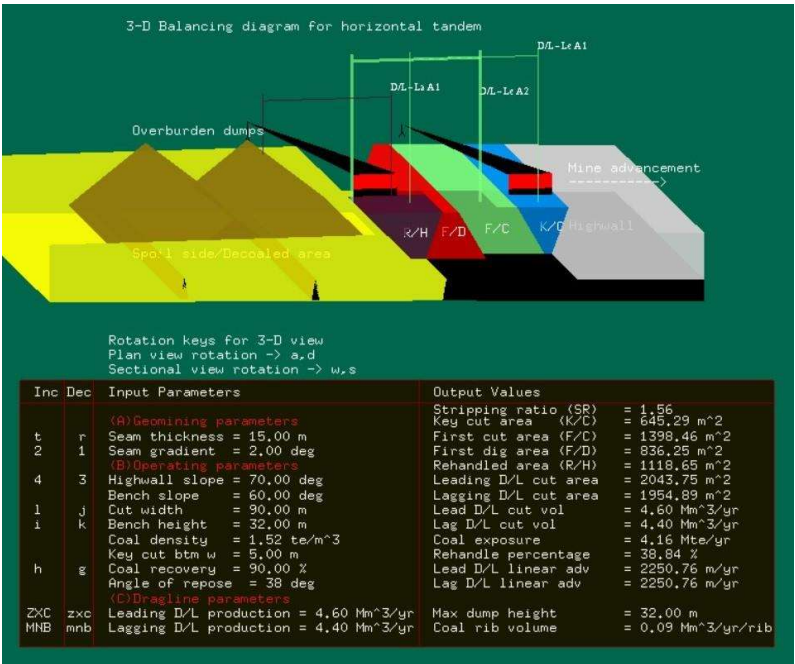


Figure 6.3.2: 3-dimensional balancing diagram for 4.60 Mm³ leading and 4.40 Mm³ lagging dragline.

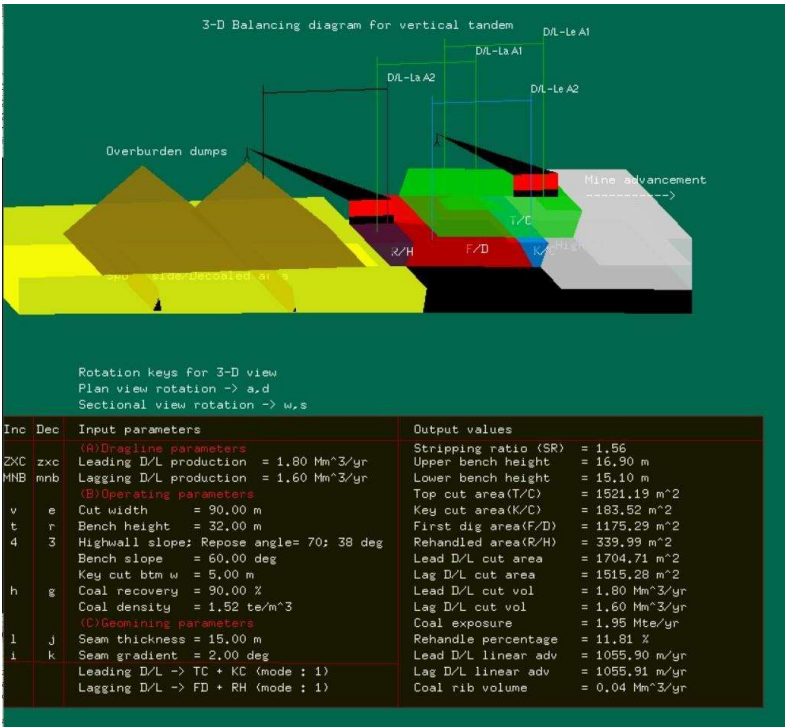


Figure 6.3.3: 3-dimensional balancing diagram for 1.80 Mm³ leading and 1.60 Mm³ lagging dragline.

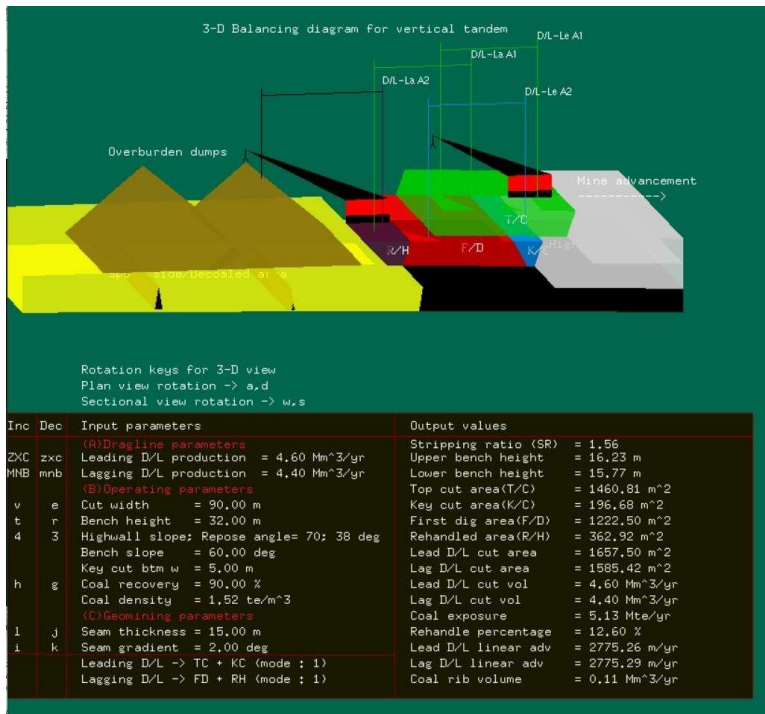


Figure 6.3.4: 3-dimensional balancing diagram for 4.60 Mm³ leading and 4.40 Mm³ lagging dragline.

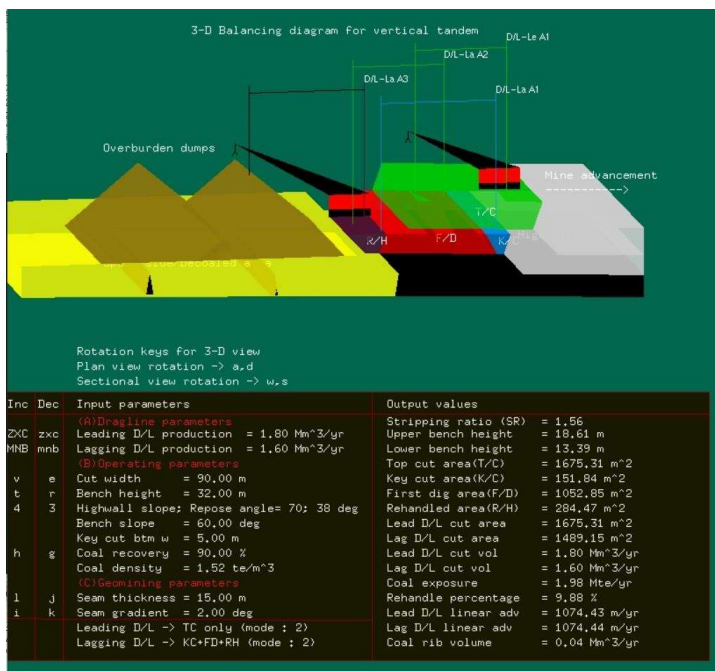


Figure 6.3.5: 3-dimensional balancing diagram for 1.80 Mm³ leading and 1.60 Mm³ lagging dragline.

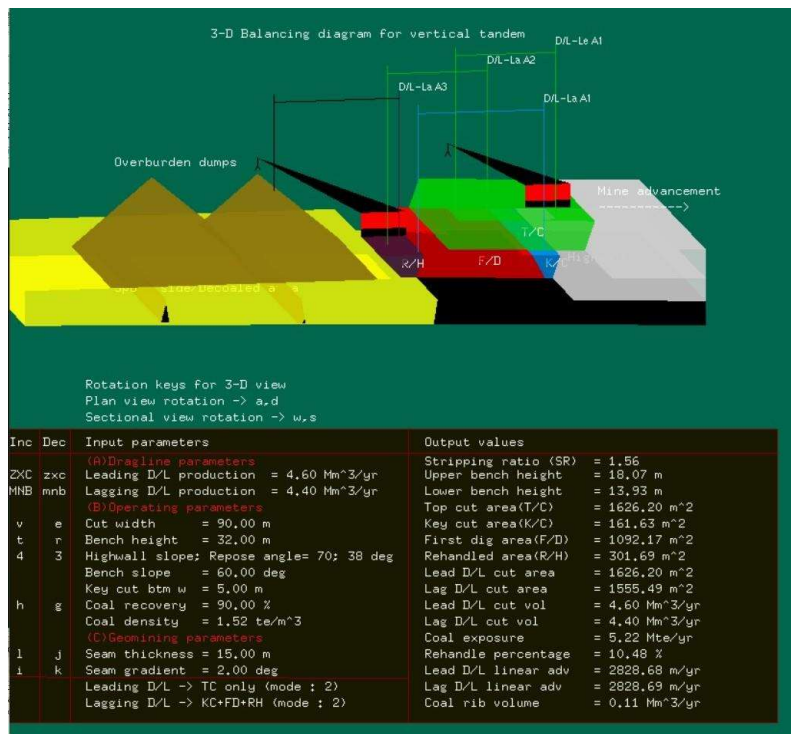


Figure 6.3.6: 3-dimensional balancing diagram for 4.60 Mm³ leading and 4.40 Mm³ lagging dragline.

6.4 Relationship equations Case- I. Le fix La vary.

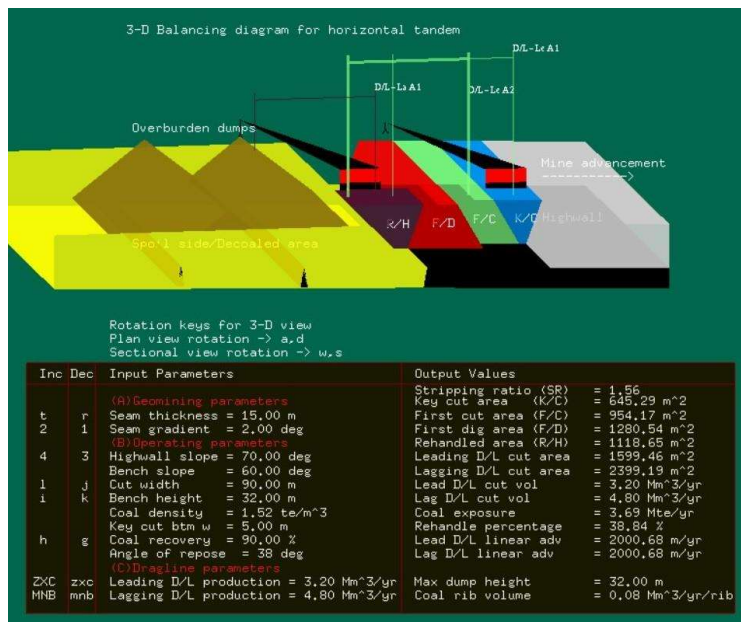


Figure 6.4.2: 3-dimensional balancing diagram for 3.20 Mm³ leading and 4.80 Mm³ lagging dragline.

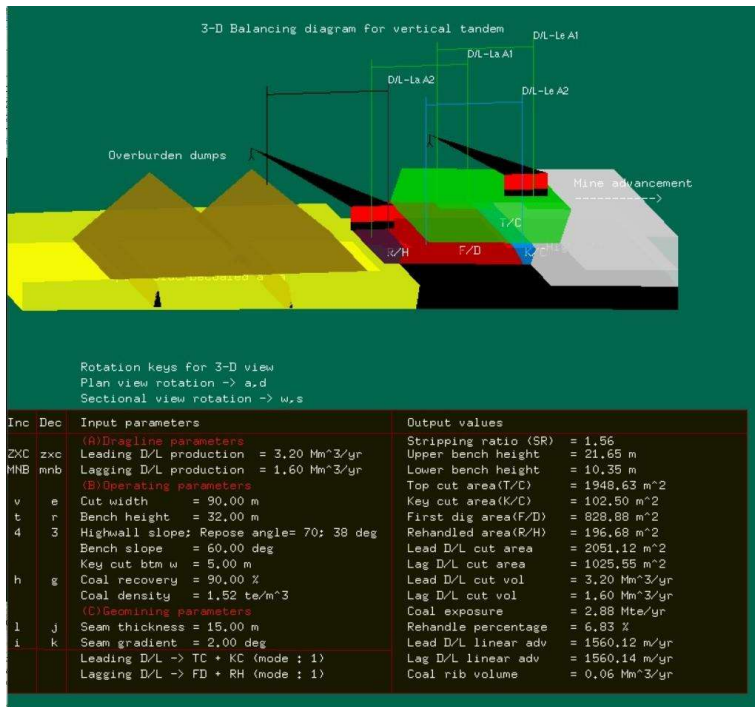


Figure 6.4.3: 3-dimensional balancing diagram for 3.20 Mm³ leading and 1.60 Mm³ lagging dragline.

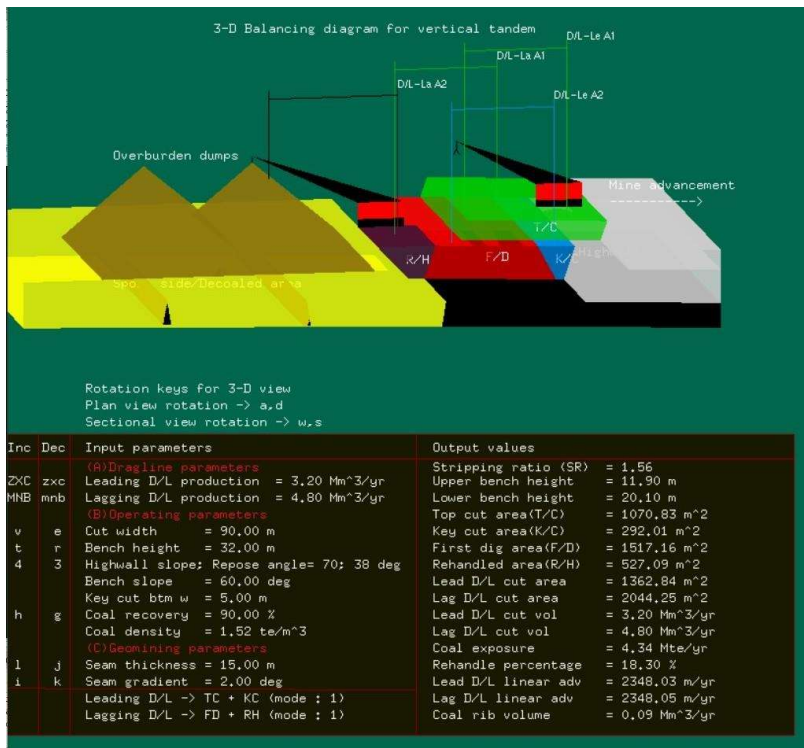


Figure 6.4.4: 3-dimensional balancing diagram for 3.20 Mm³ leading and 4.80 Mm³ lagging dragline.

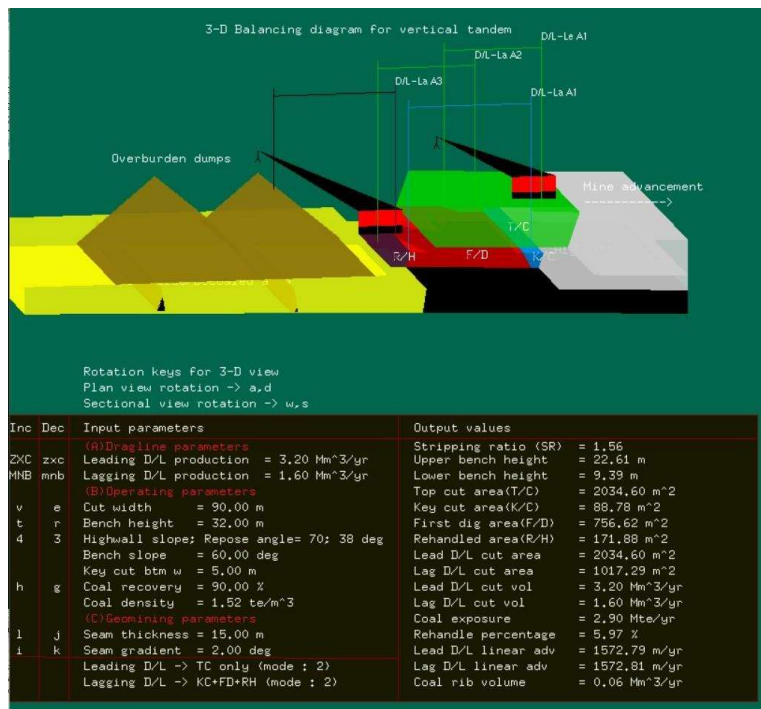


Figure 6.4.5: 3-dimensional balancing diagram for 3.20 Mm³ leading and 1.60 Mm³ lagging dragline.

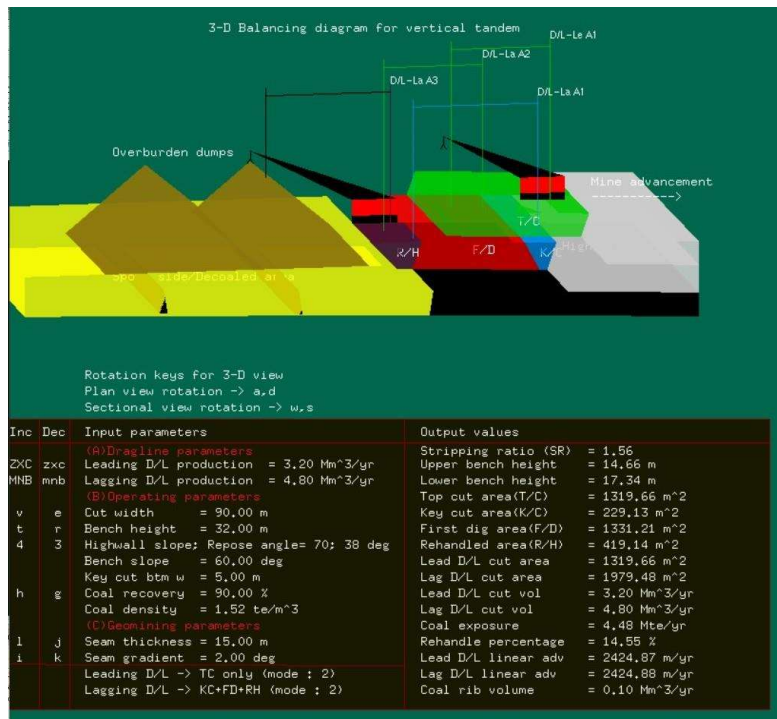


Figure 6.4.6: 3-dimensional balancing diagram for 3.20 Mm³ leading and 4.80 Mm³ lagging dragline.

6.4 Relationship equations Case- II. Le vary La fix

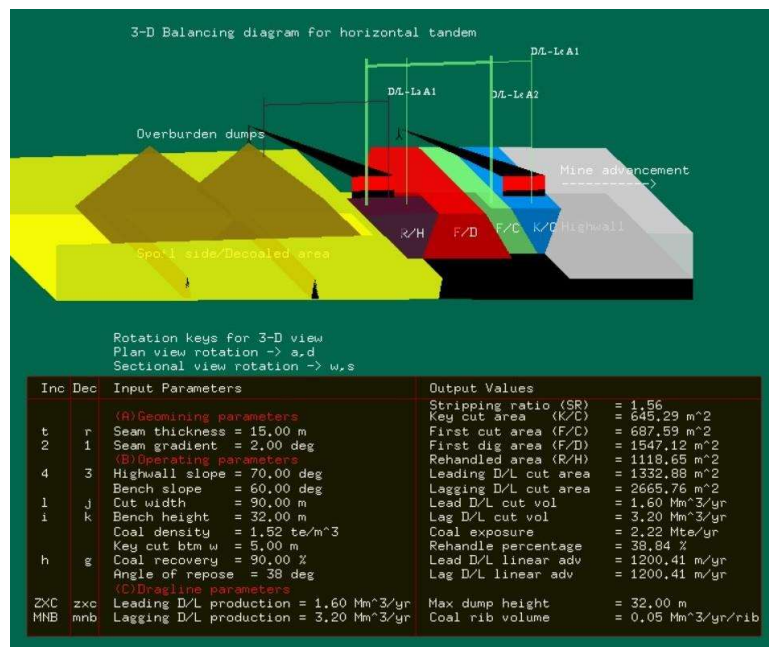


Figure 6.4.7: 3-dimensional balancing diagram for 1.60 Mm³ leading and 3.20 Mm³ lagging dragline.

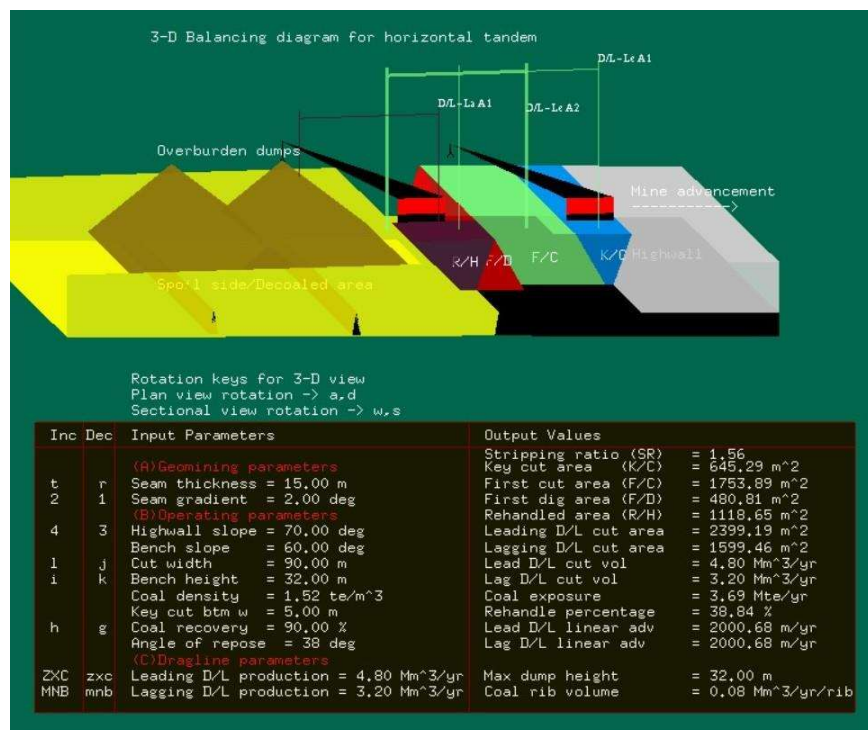


Figure 6.4.8: 3-dimensional balancing diagram for 4.80 Mm³ leading and 3.20 Mm³ lagging dragline.

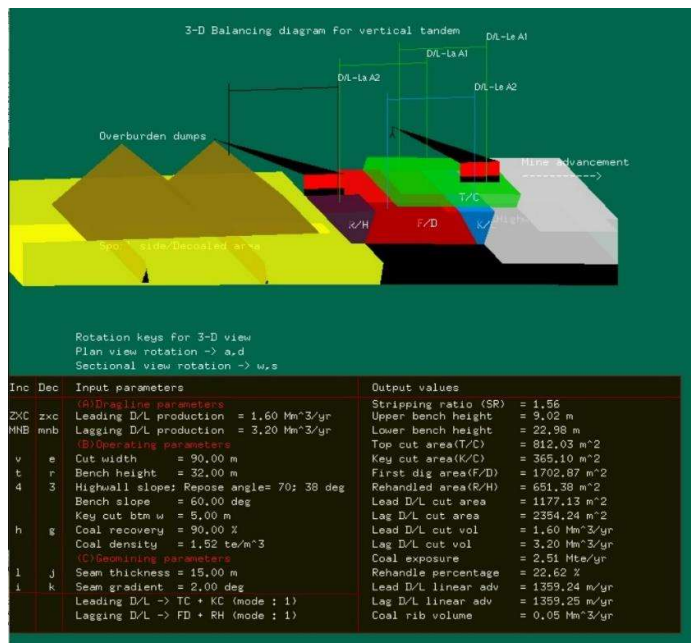


Figure 6.4.9: 3-dimensional balancing diagram for 1.60 Mm³ leading and 3.20 Mm³ lagging dragline.

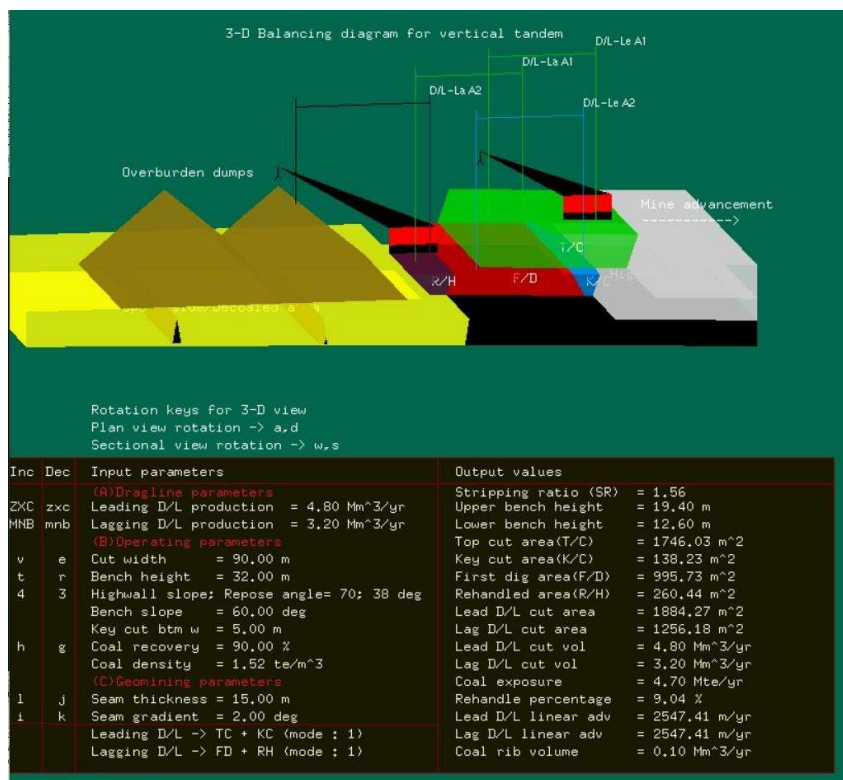


Figure 6.4.10: 3-dimensional balancing diagram for 4.80 Mm³ leading and 3.20 Mm³ lagging dragline.

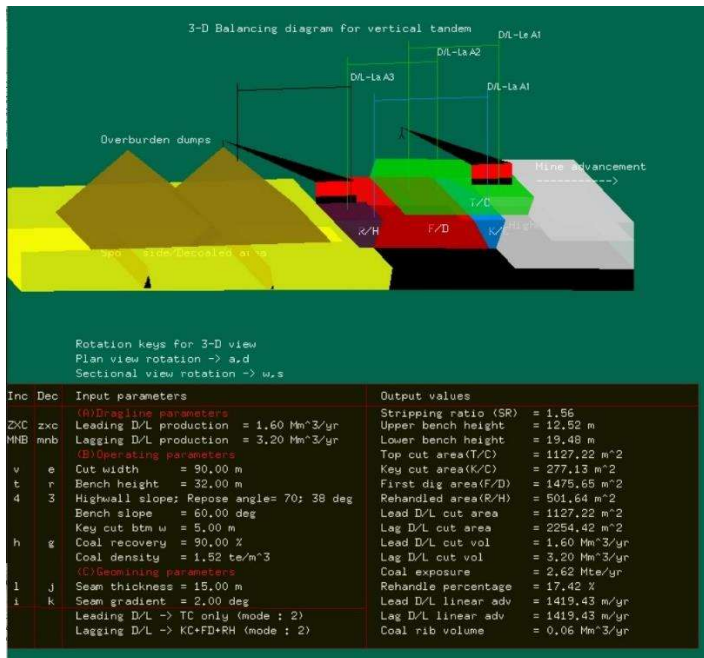


Figure 6.4.11: 3-dimensional balancing diagram for 1.60 Mm³ leading and 3.20 Mm³ lagging dragline.

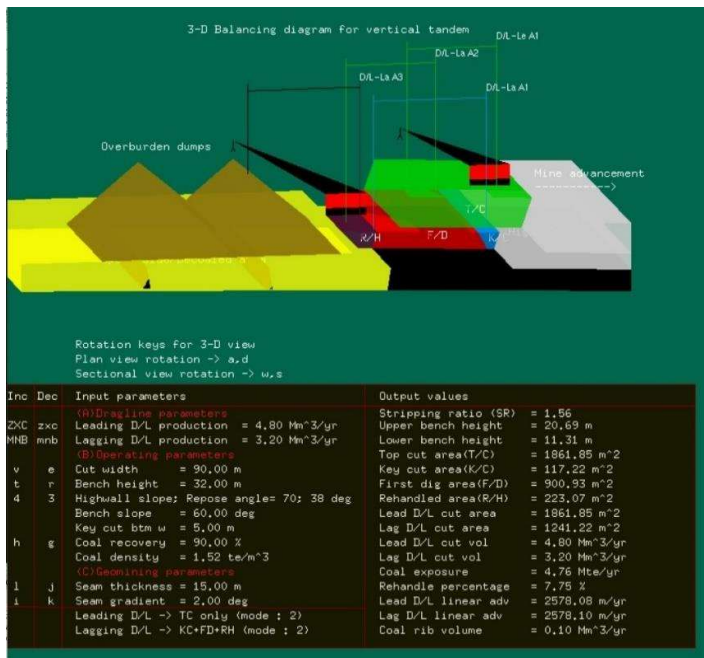


Figure 6.4.12: 3-dimensional balancing diagram for 4.80 Mm³ leading and 3.20 Mm³ lagging dragline.

6.5 Influence of cut width.

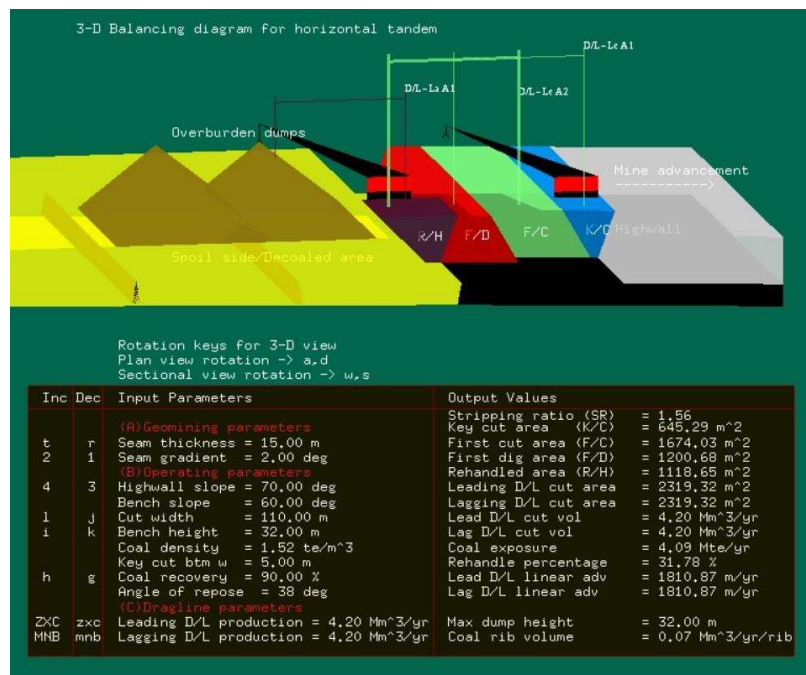


Figure 6.5.2: 3-dimensional balancing diagram for 110 m cut width.

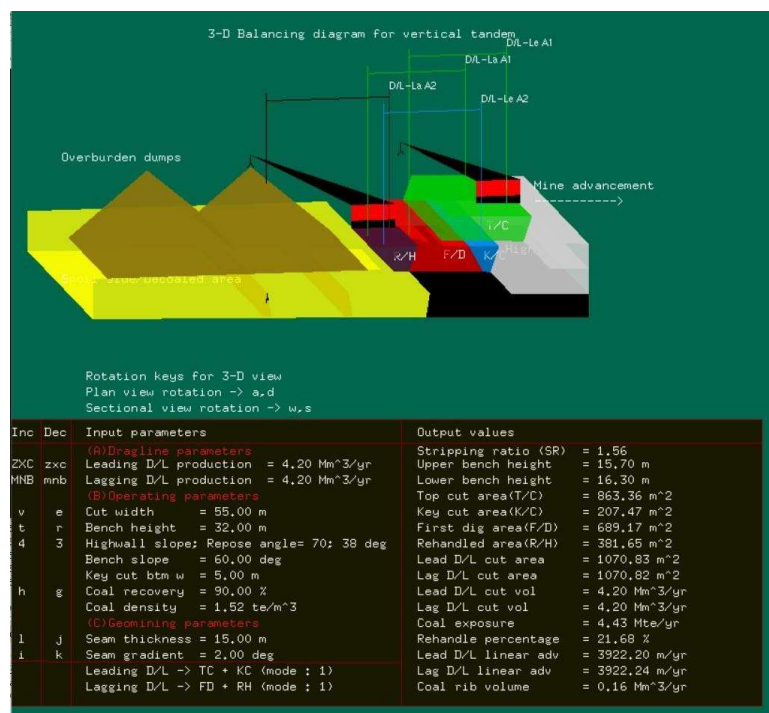


Figure 6.5.3: 3-dimensional balancing diagram for 55 m cut width.

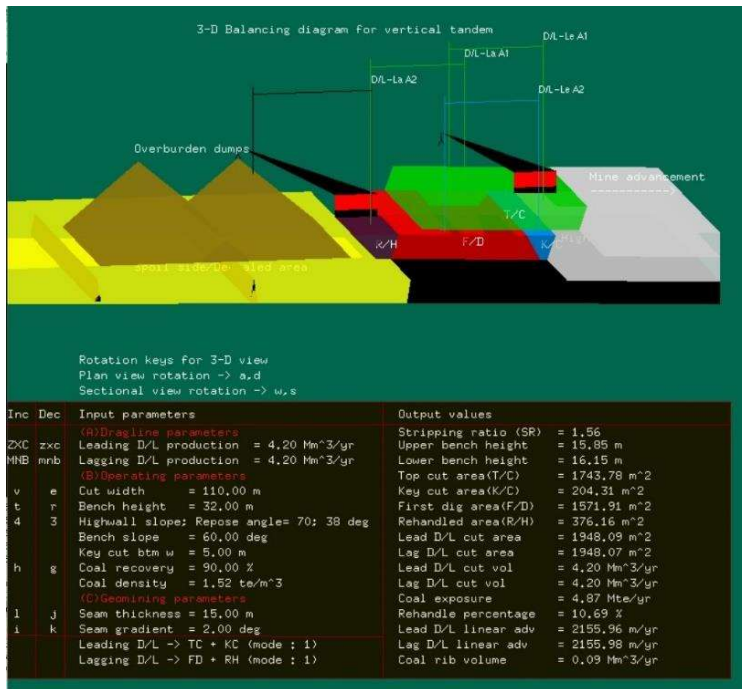


Figure 6.5.4: 3-dimensional balancing diagram for 110 m cut width.

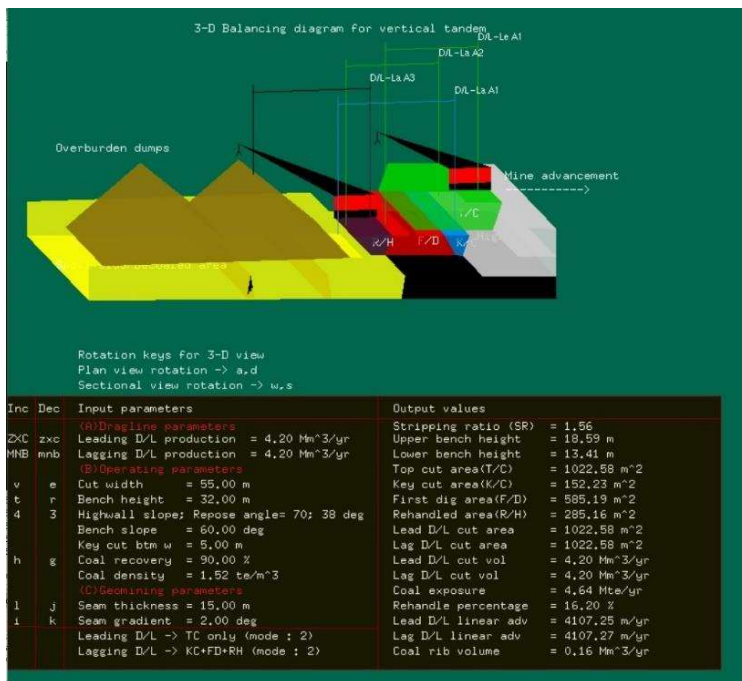


Figure 6.5.5: 3-dimensional balancing diagram for 55 m cut width.

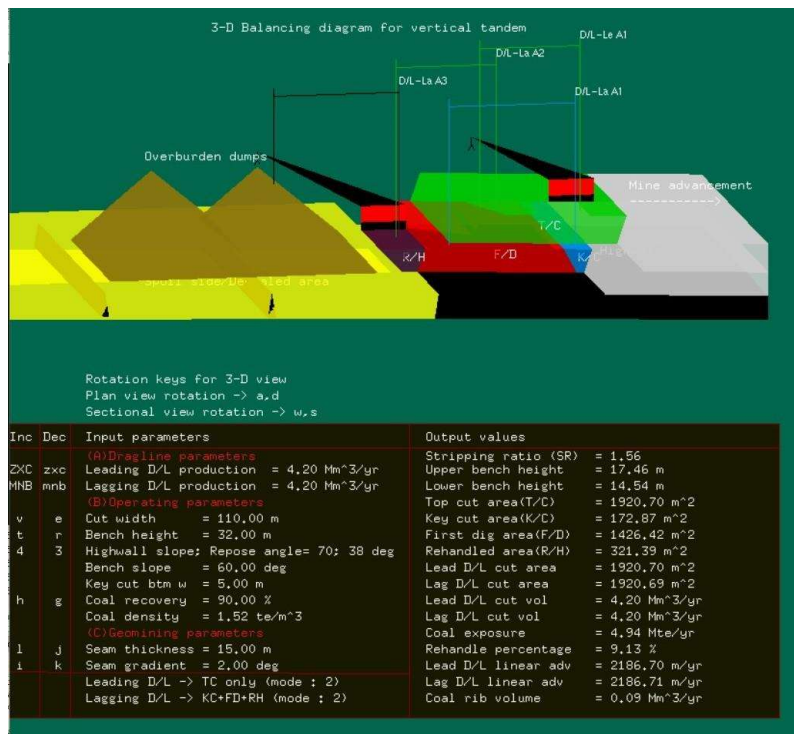


Figure 6.5.6: 3-dimensional balancing diagram for 110 m cut width.

6.6 Influence of stripping bench height

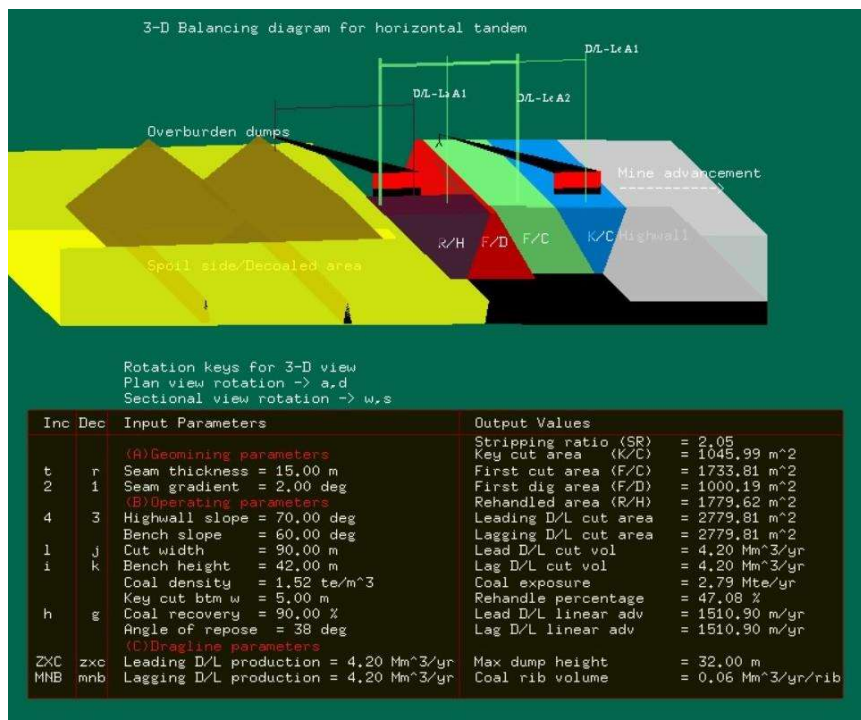


Figure 6.6.2: 3-dimensional balancing diagram for 42 m bench height.

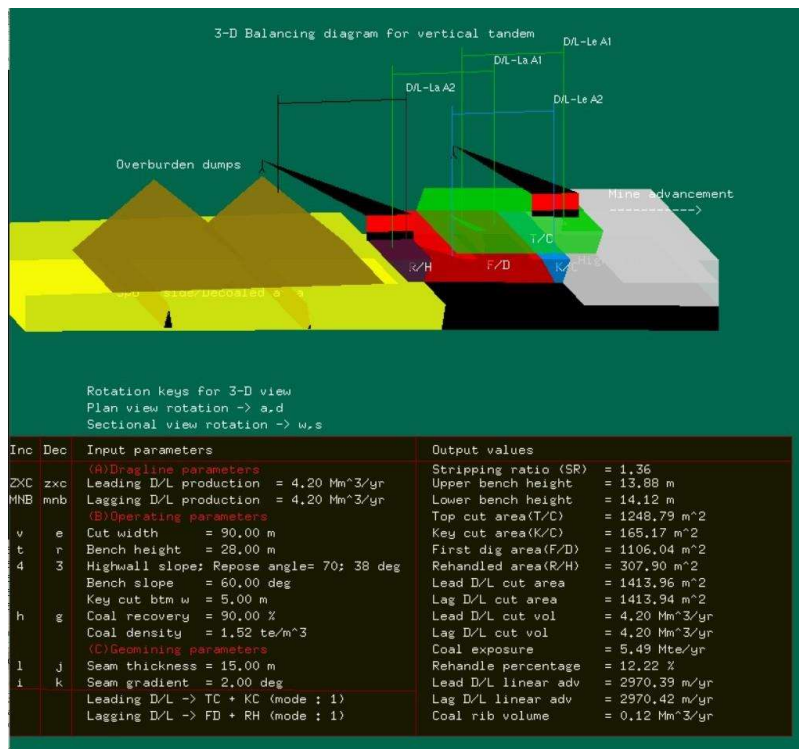


Figure 6.6.3: 3-dimensional balancing diagram for 28 m bench height.

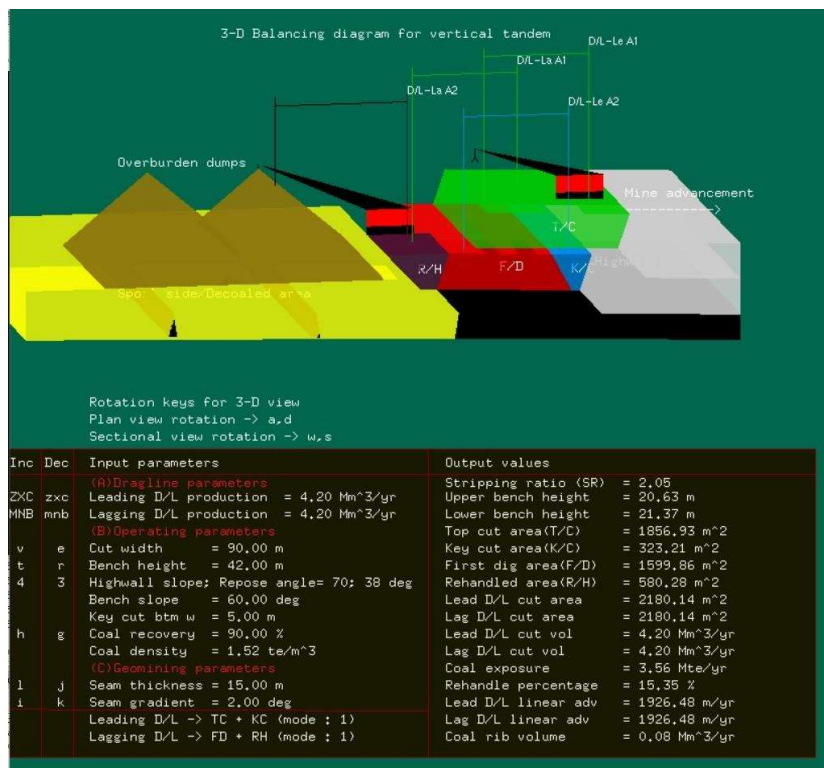


Figure 6.6.4: 3-dimensional balancing diagram for 42 m bench height.

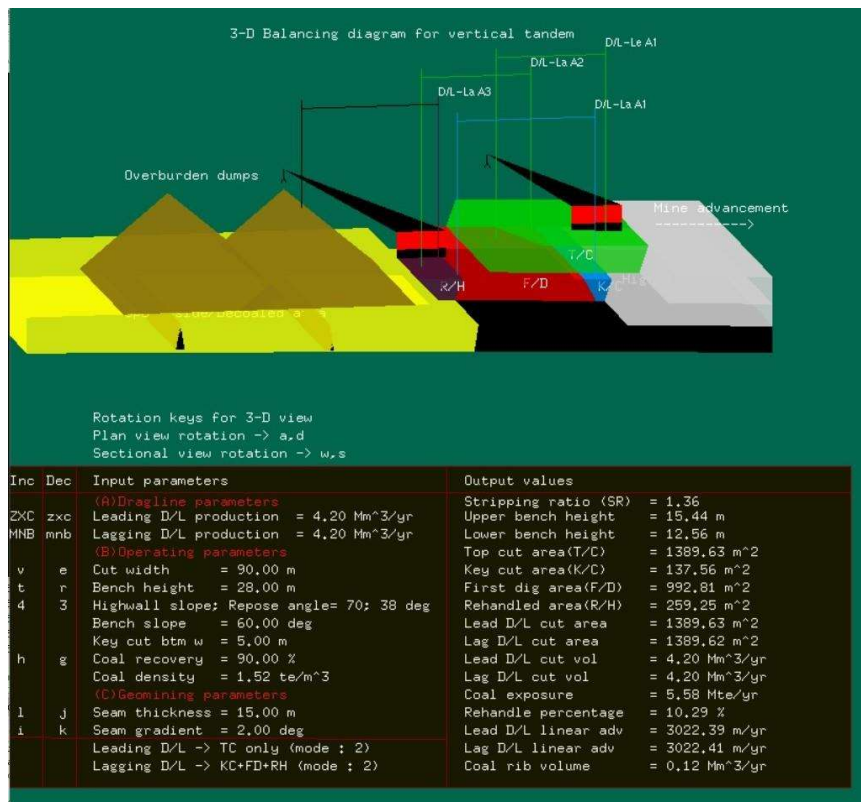


Figure 6.6.5: 3-dimensional balancing diagram for 28 m bench height.

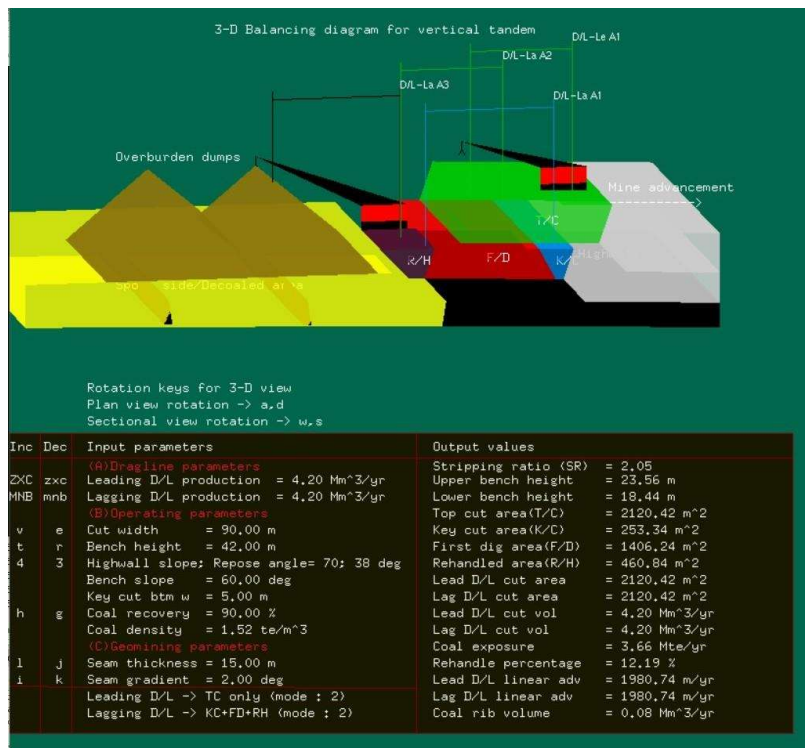


Figure 6.6.6: 3-dimensional balancing diagram for 42 m bench height.

6.7 Influence of coal seam thickness

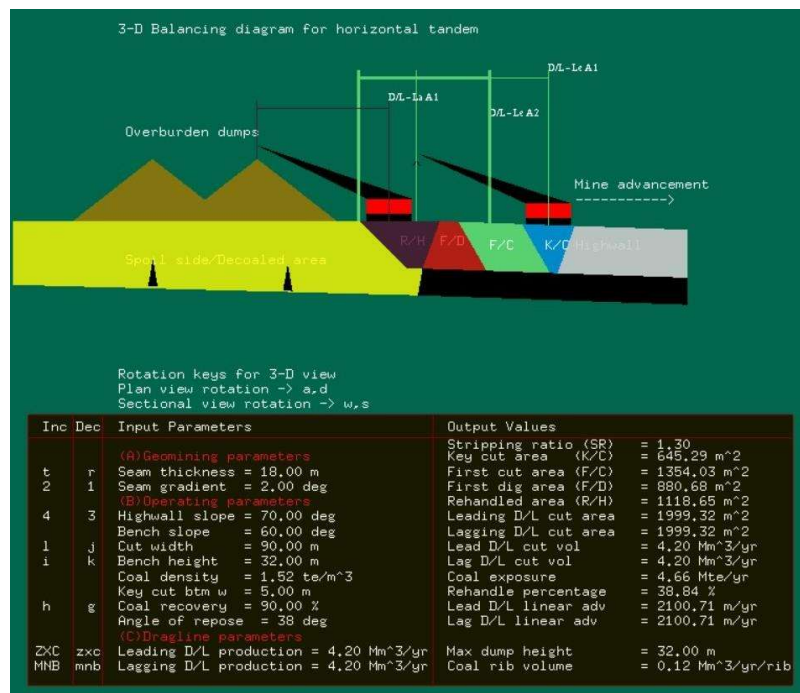


Figure 6.7.2: 3-dimensional balancing diagram for 18 m thick coal seam.

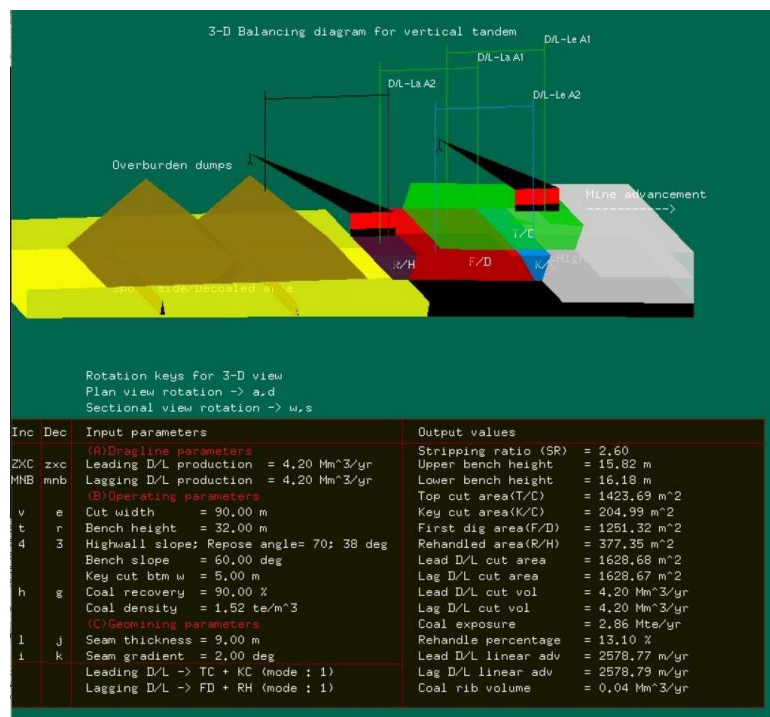


Figure 6.7.3: 3-dimensional balancing diagram for 9 m thick coal seam.

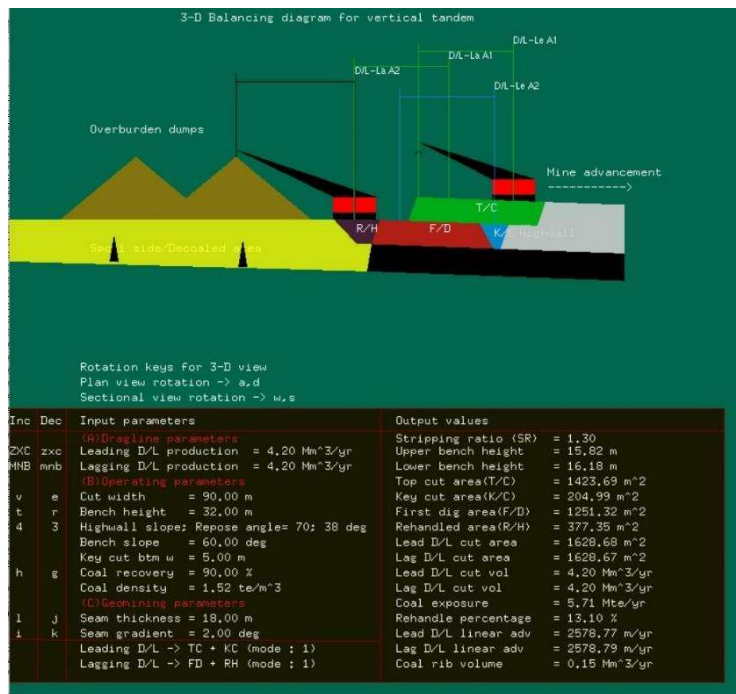


Figure 6.7.4: 3-dimensional balancing diagram for 18 m thick coal seam.

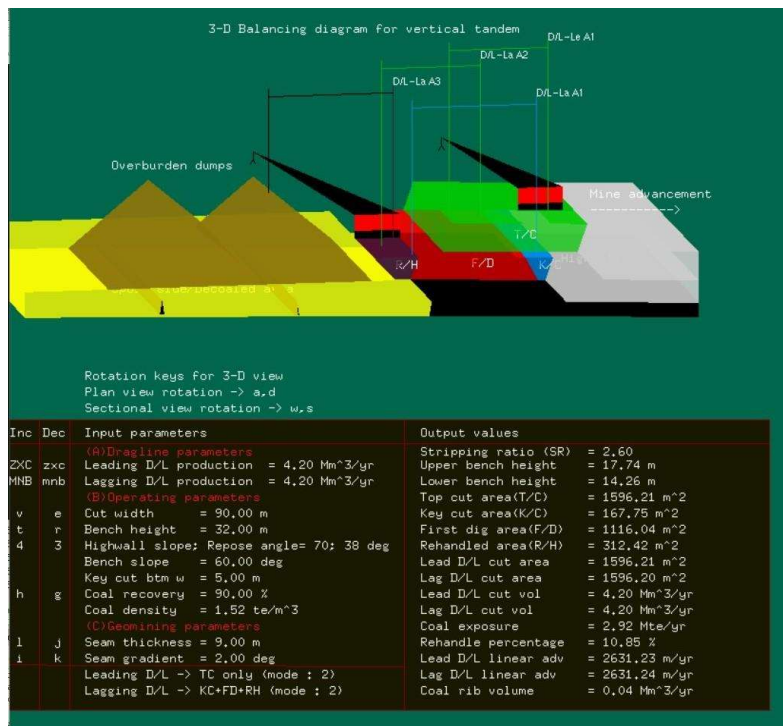


Figure 6.7.5: 3-dimensional balancing diagram for 9 m thick coal seam.

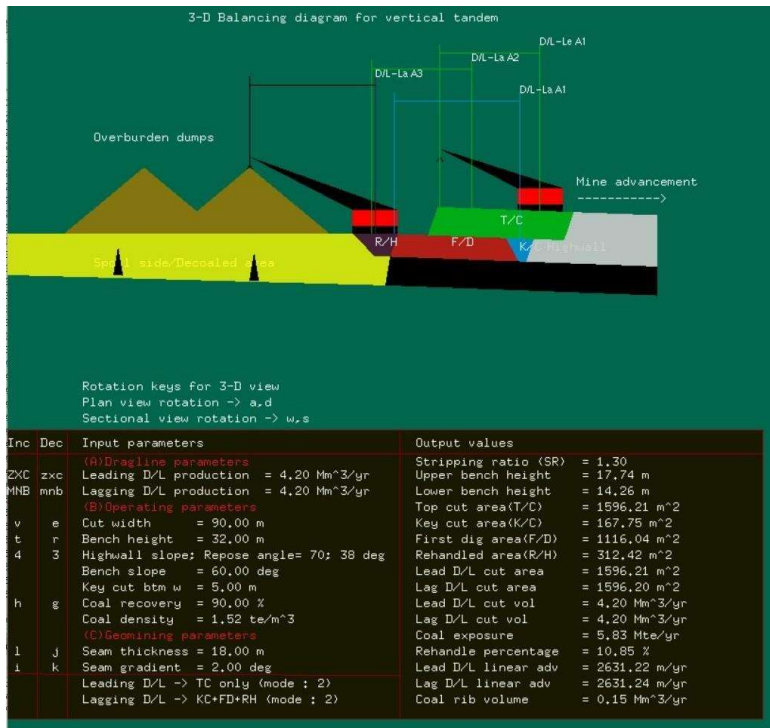


Figure 6.7.6: 3-dimensional balancing diagram for 18 m thick coal seam.

6.8 Influence of seam gradient

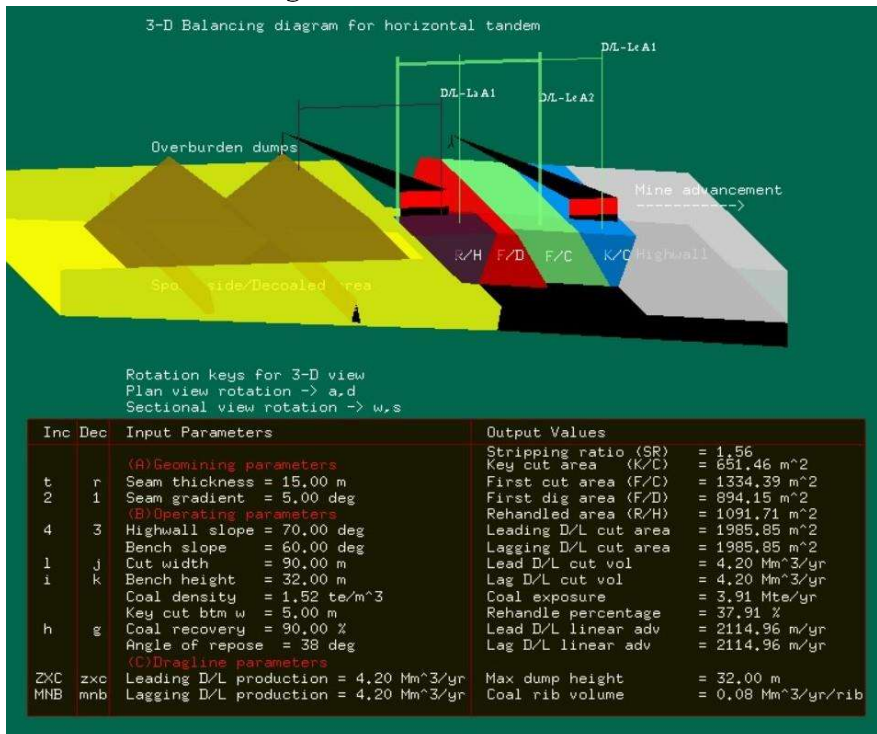


Figure 6.8.2: 3-dimensional balancing diagram for coal seam inclination of 5 degree.

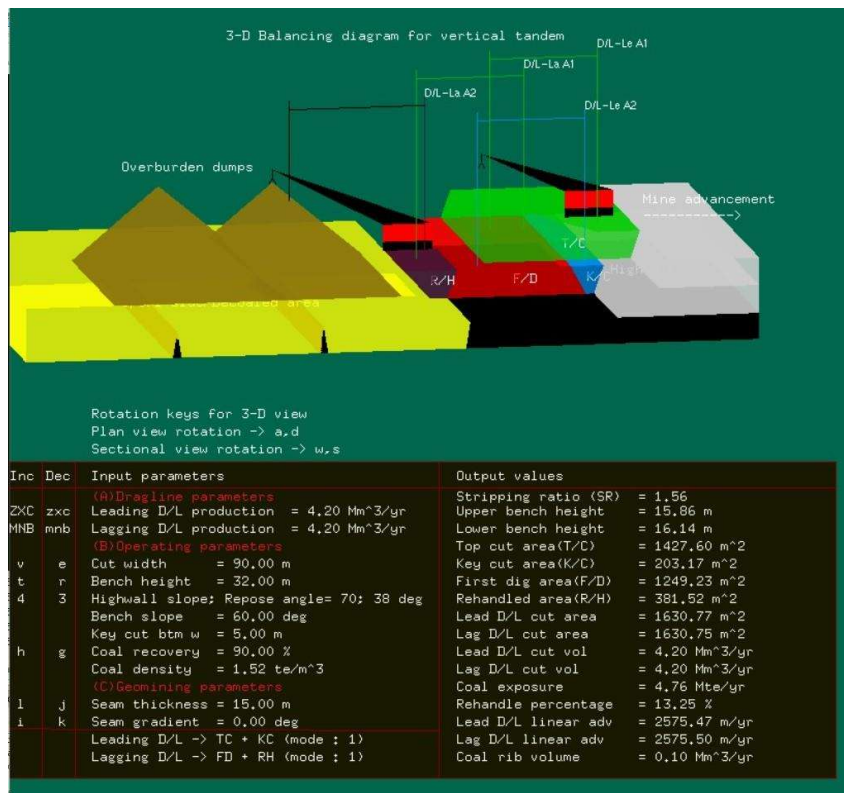


Figure 6.8.3: 3-dimensional balancing diagram for coal seam inclination of 0 degree.

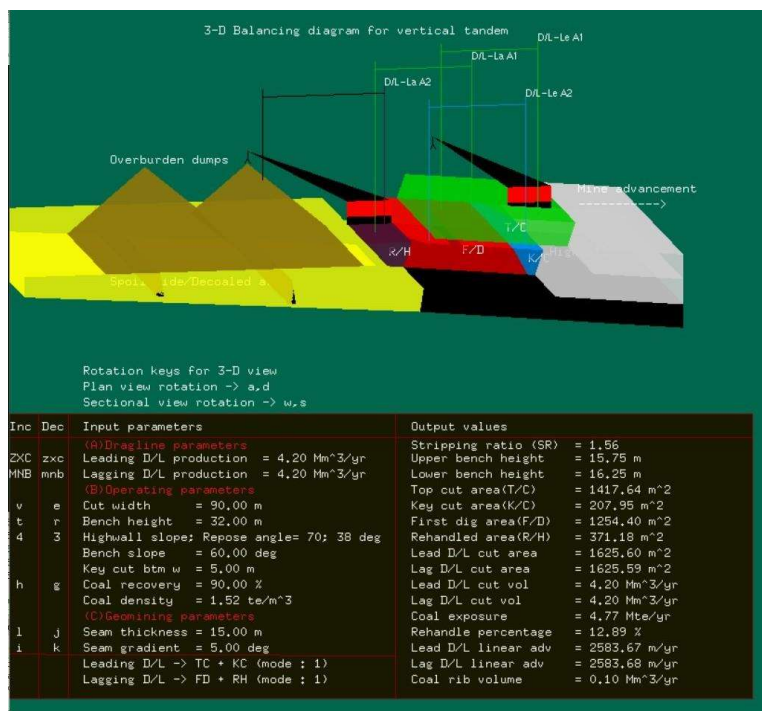


Figure 6.8.4: 3-dimensional balancing diagram for coal seam inclination of 5 degree.

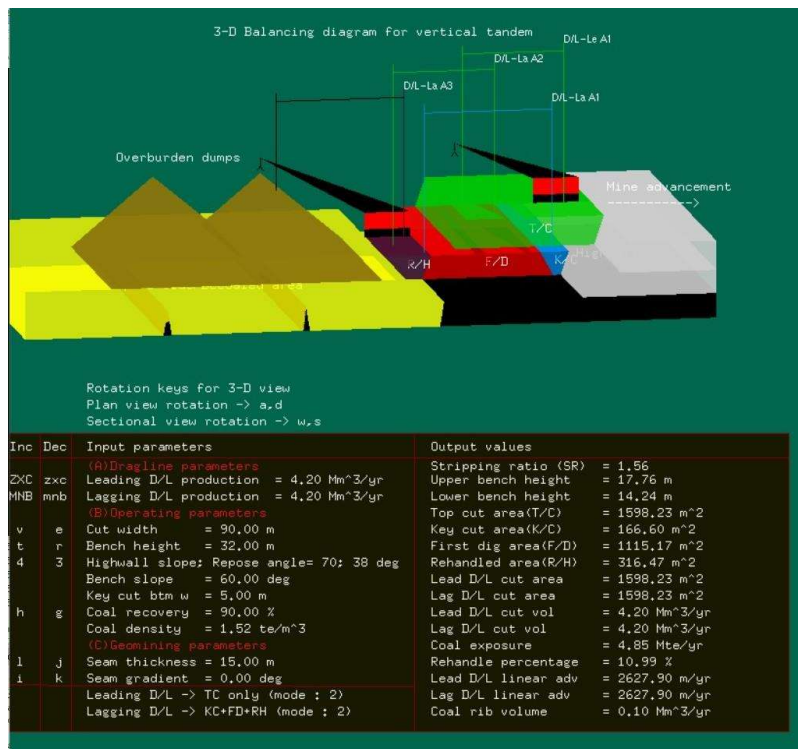


Figure 6.8.5: 3-dimensional balancing diagram for coal seam inclination of 0 degree.

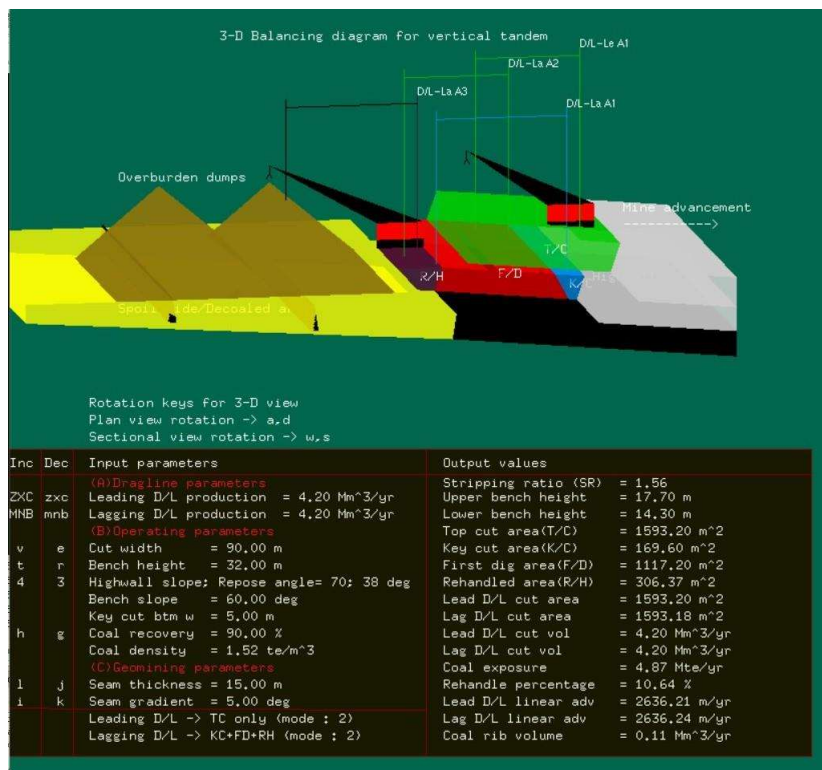


Figure 6.8.6: 3-dimensional balancing diagram for coal seam inclination of 5 degree.

6.9 To find the best combination of cut width and bench height within the possible working range of draglines.

Case 1: Horizontal tandem

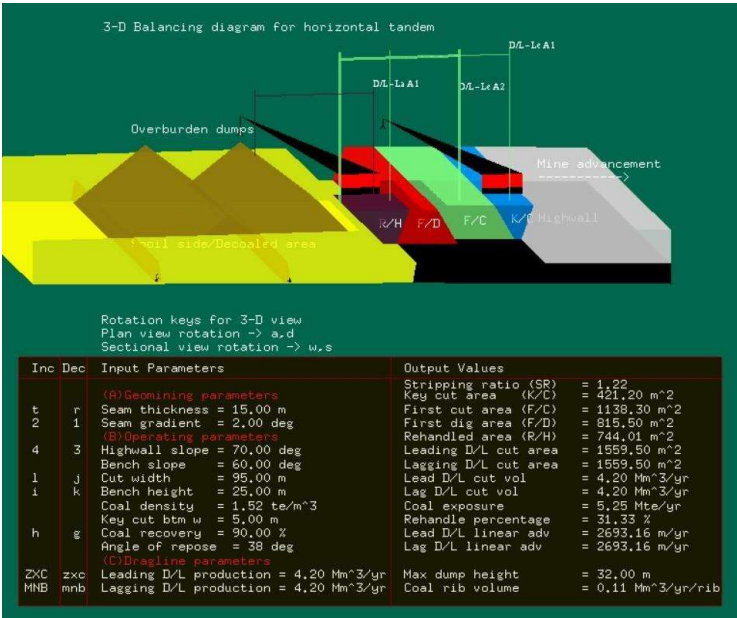


Figure 6.9.1: 3-dimensional balancing diagram for cut width of 95 m and bench height of 25 m.

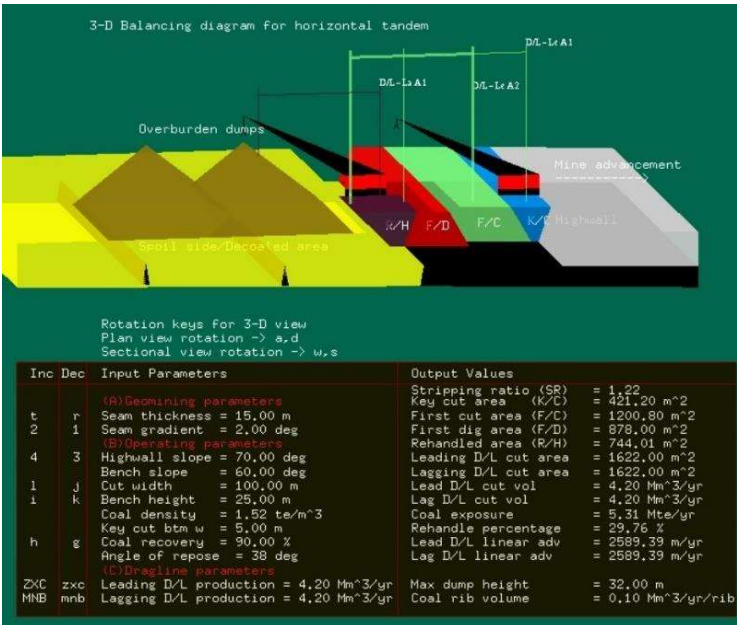


Figure 6.9.2: 3-dimensional balancing diagram for cut width of 100 m and bench height of 25 m.

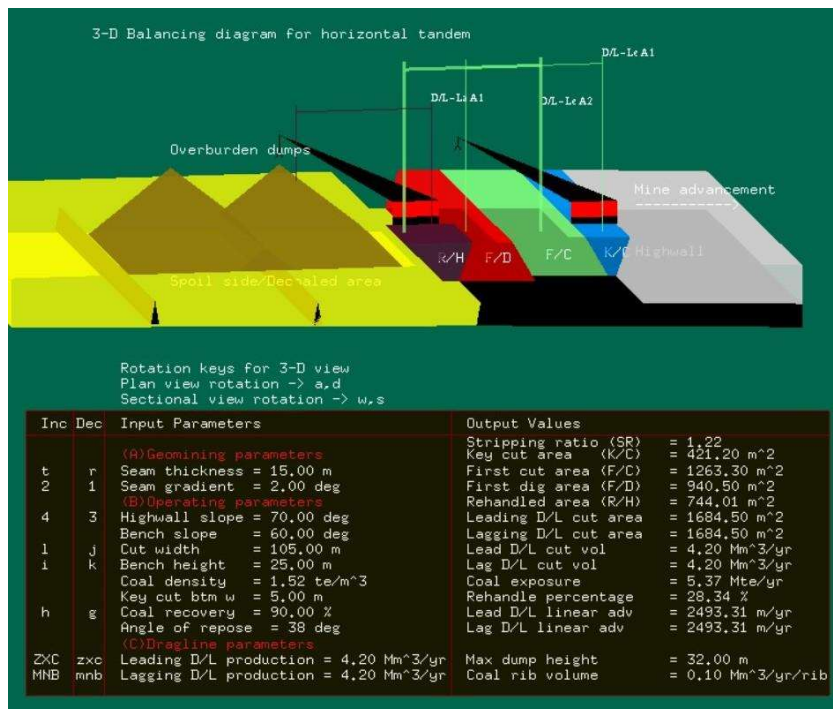


Figure 6.9.3: 3-dimensional balancing diagram for cut width of 105 m and bench height of 25 m.

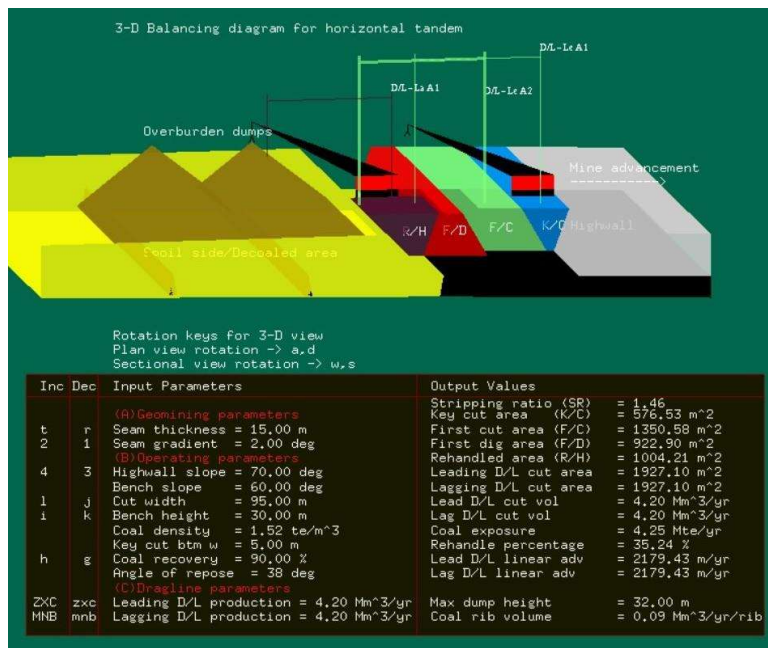


Figure 6.9.4: 3-dimensional balancing diagram for cut width of 95 m and bench height of 30 m.

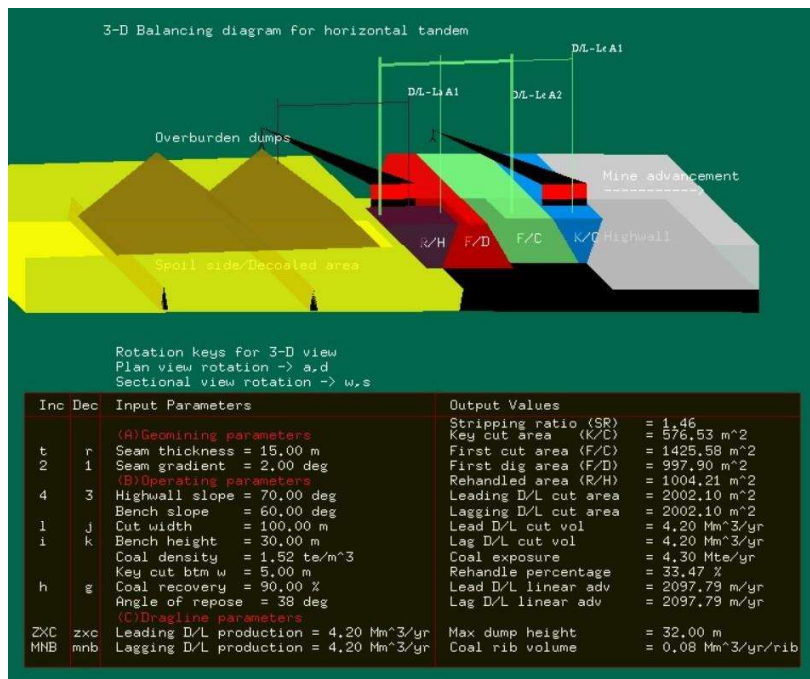


Figure 6.9.5: 3-dimensional balancing diagram for cut width of 100 m and bench height of 30 m.

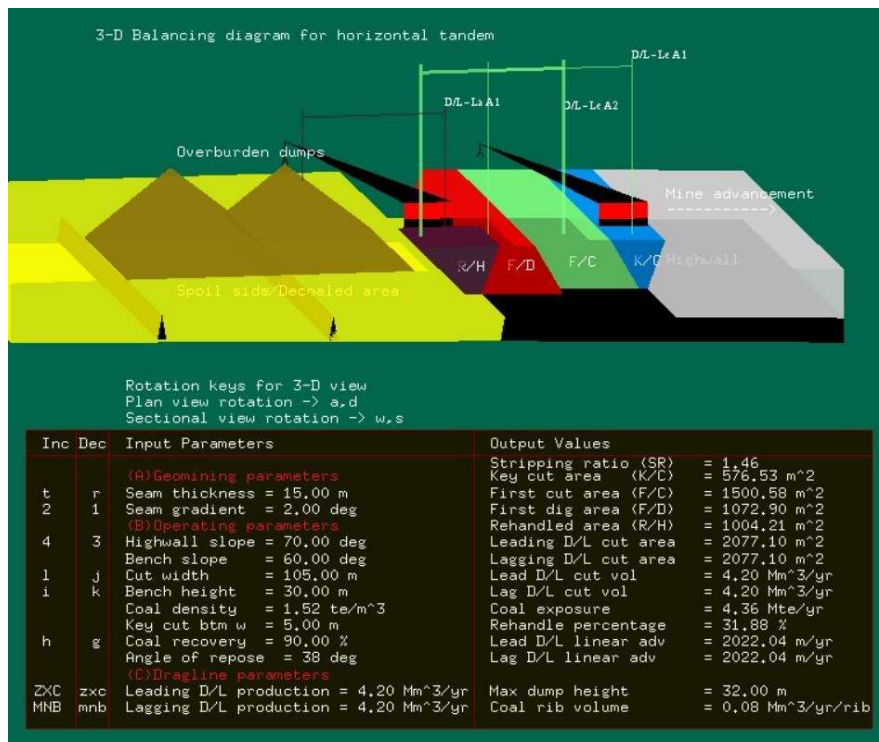


Figure 6.9.6: 3-dimensional balancing diagram for cut width of 105 m and bench height of 30 m.

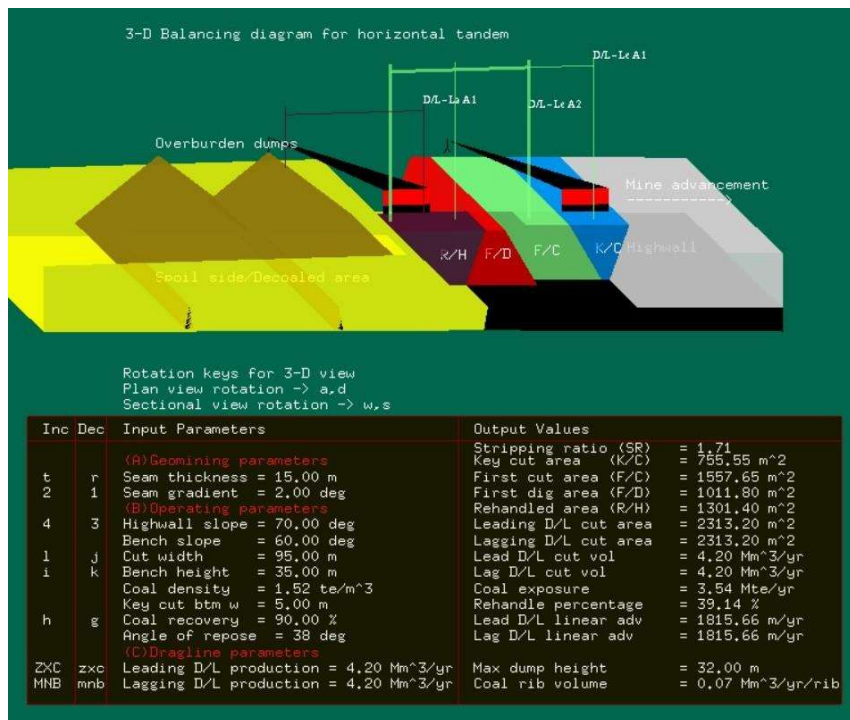


Figure 6.9.7: 3-dimensional balancing diagram for cut width of 95 m and bench height of 35 m.

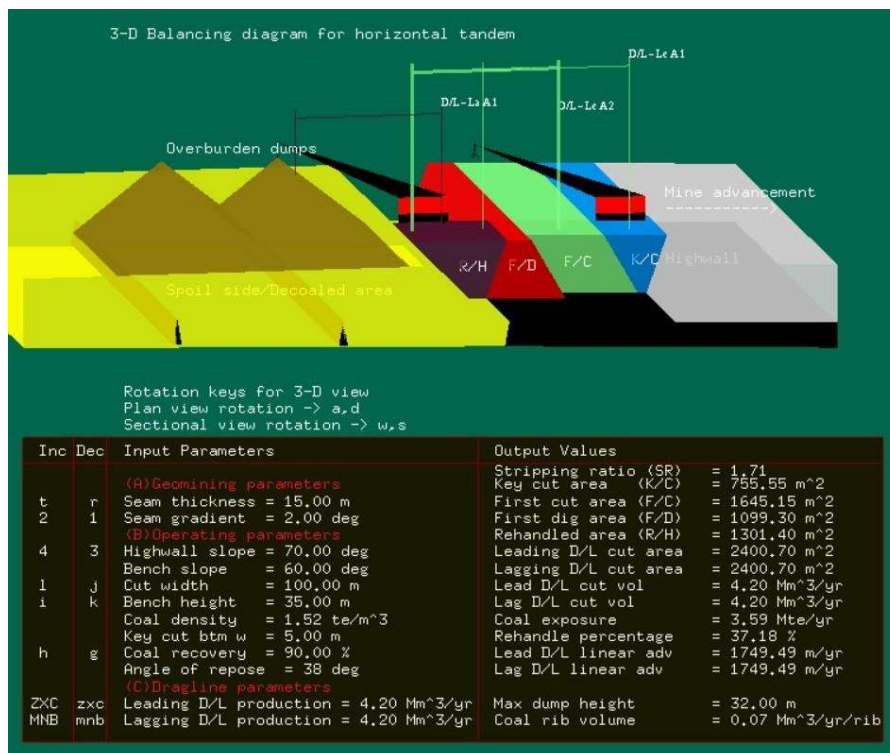


Figure 6.9.8: 3-dimensional balancing diagram for cut width of 100 m and bench height of 35 m.

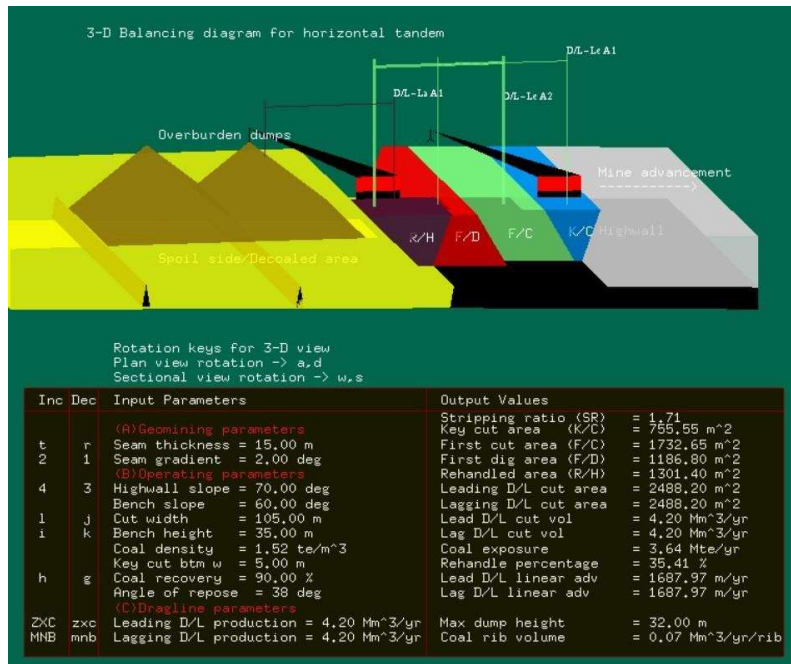


Figure 6.9.9: 3-dimensional balancing diagram for cut width of 105 m and bench height of 35 m.

6.9 - Case 2: Vertical tandem operation mode 1

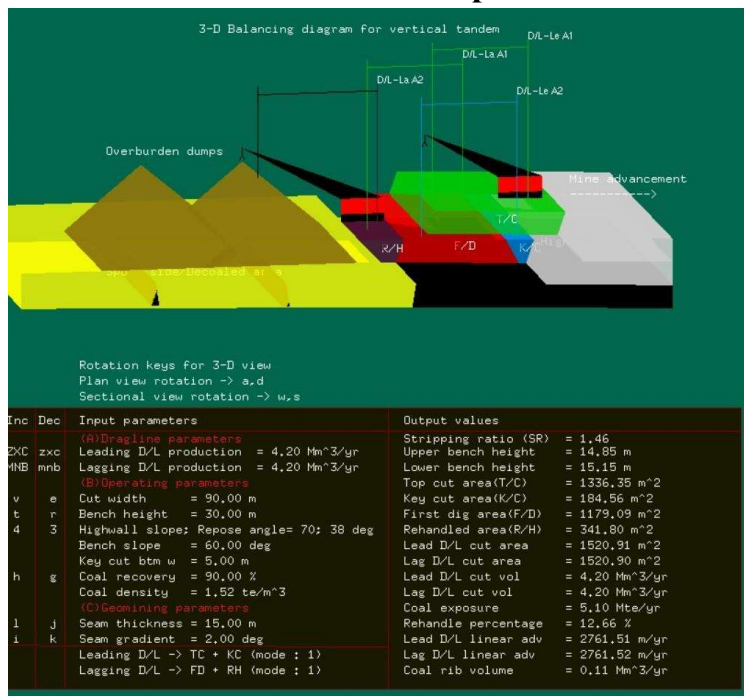


Figure 6.9.10: 3-dimensional balancing diagram for cut width of 90 m and bench height of 30 m.

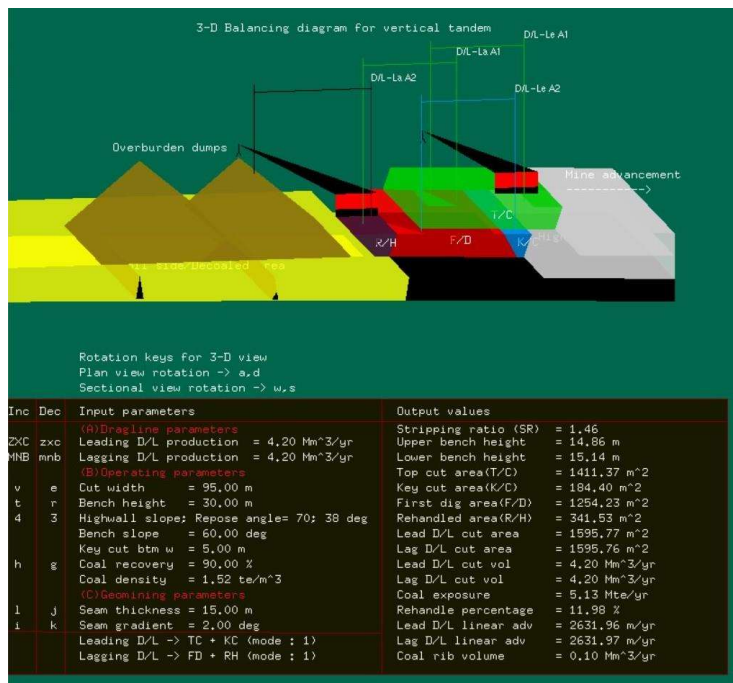


Figure 6.9.11: 3-dimensional balancing diagram for cut width of 95 m and bench height of 30 m.

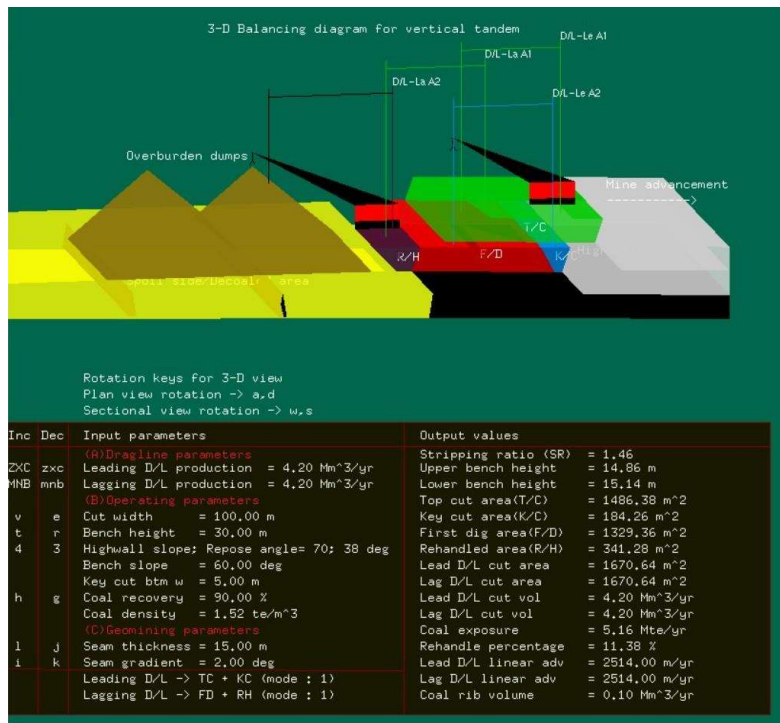


Figure 6.9.12: 3-dimensional balancing diagram for cut width of 100 m and bench height of 30 m.

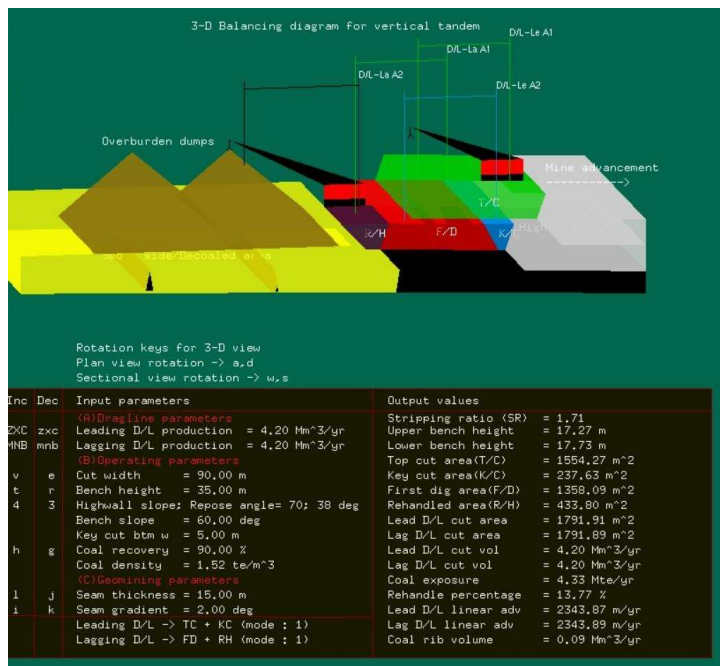


Figure 6.9.13: 3-dimensional balancing diagram for cut width of 90 m and bench height of 35 m.

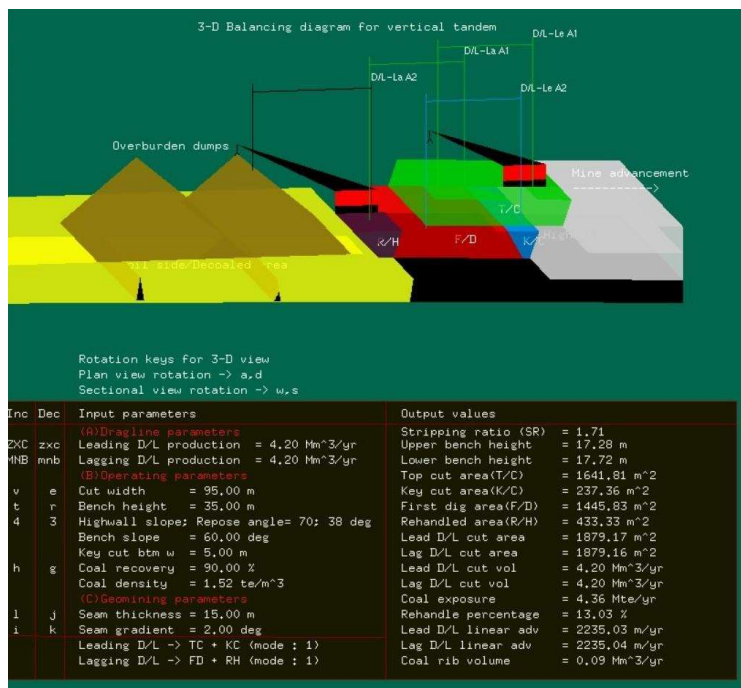


Figure 6.9.14: 3-dimensional balancing diagram for cut width of 95 m and bench height of 35 m.

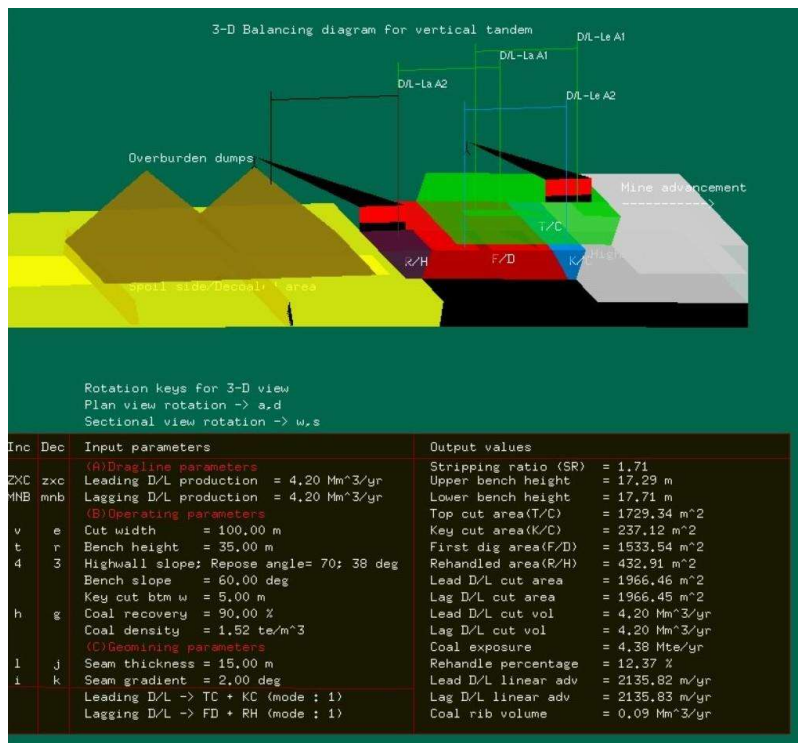


Figure 6.9.15: 3-dimensional balancing diagram for cut width of 100 m and bench height of 35 m.

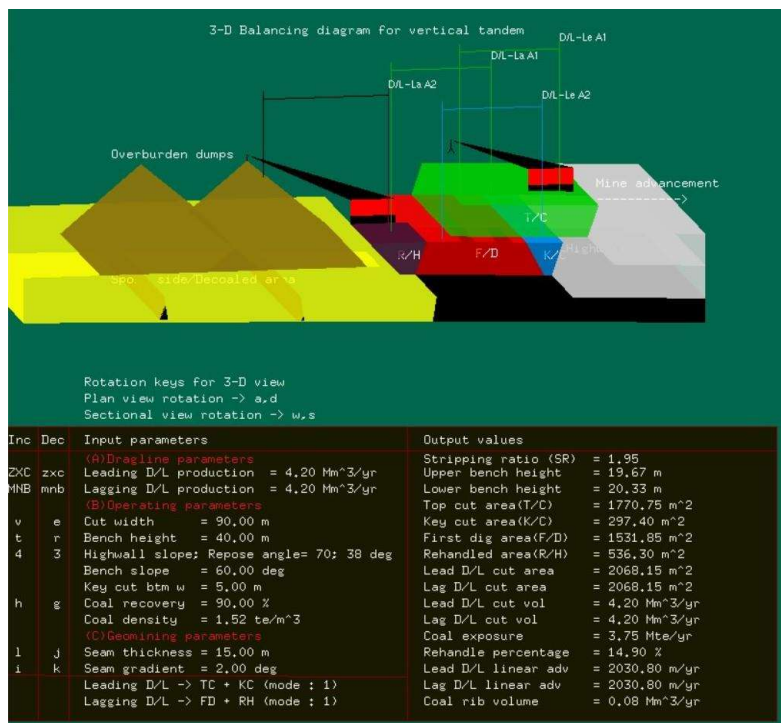


Figure 6.9.16: 3-dimensional balancing diagram for cut width of 90 m and bench height of 40 m.

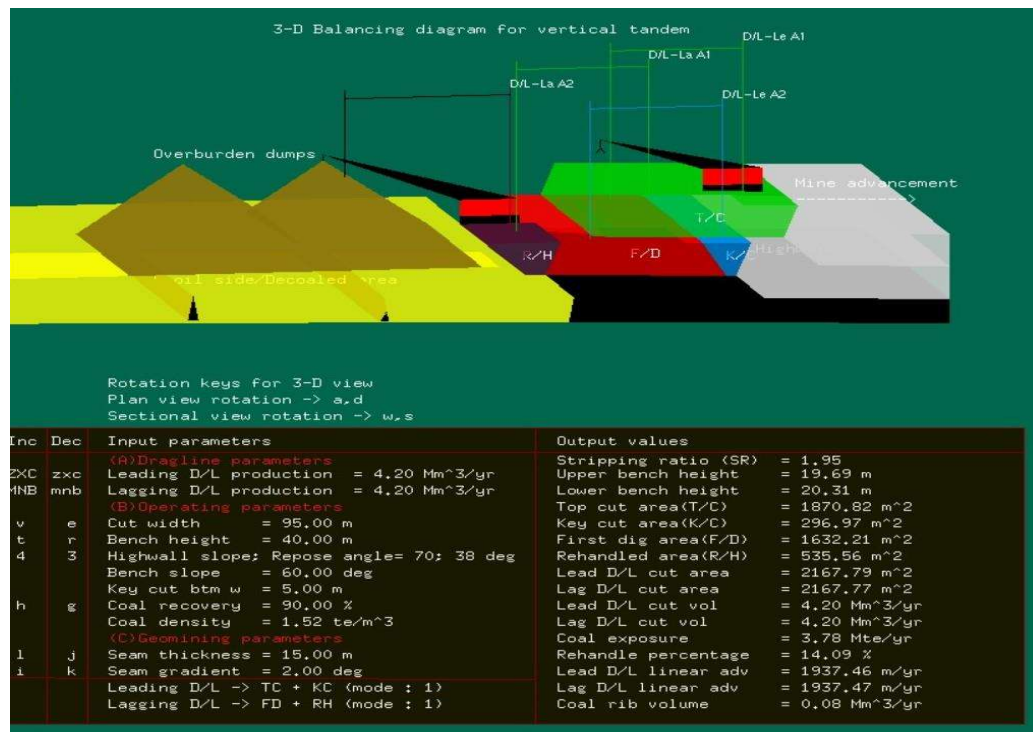


Figure 6.9.17: 3-dimensional balancing diagram for cut width of 95 m and bench height of 40 m.

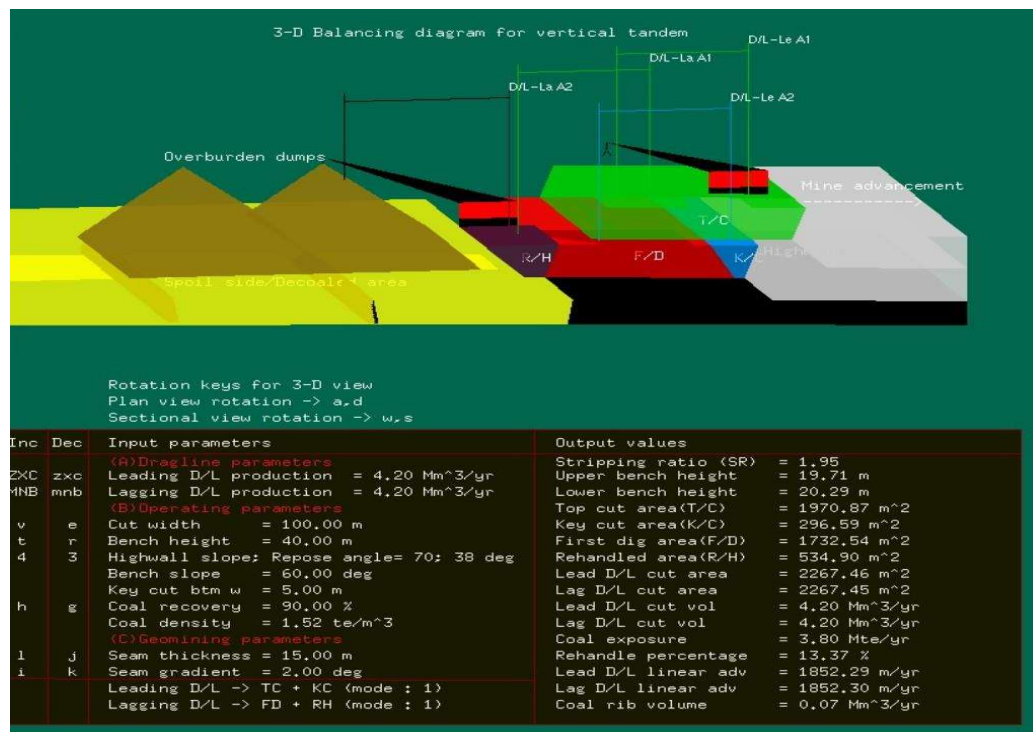


Figure 6.9.18: 3-dimensional balancing diagram for cut width of 100 m and bench height of 40 m.

6.9 - Case 3: Vertical tandem operation mode 2

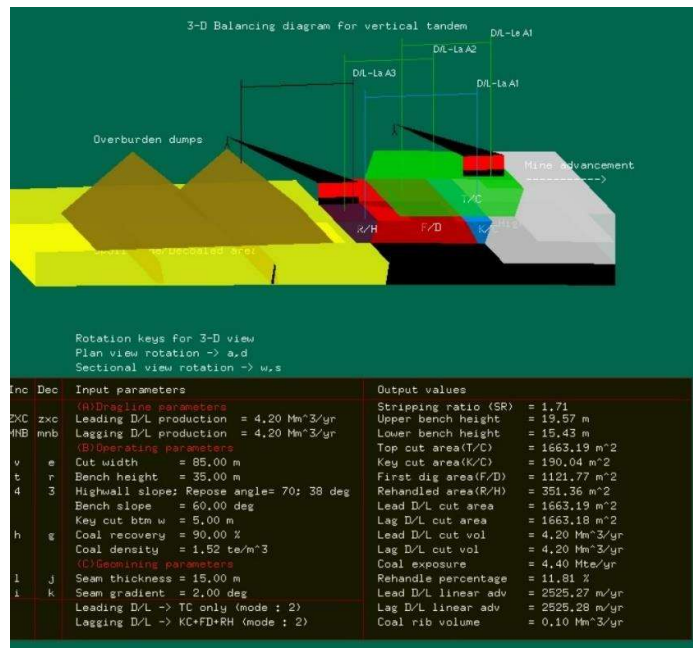


Figure 6.9.19: 3-dimensional balancing diagram for cut width of 85 m and bench height of 35 m.

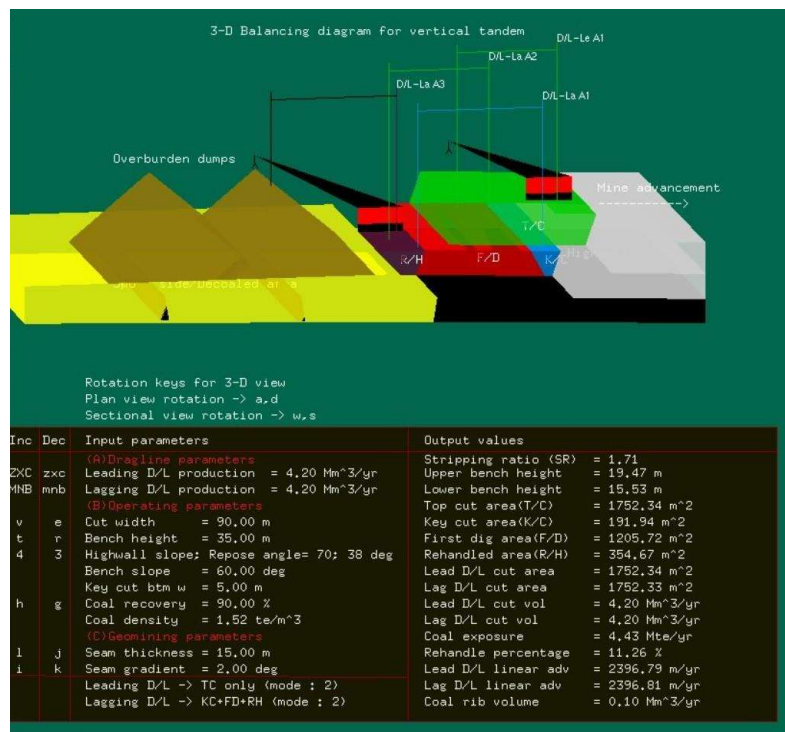


Figure 6.9.20: 3-dimensional balancing diagram for cut width of 90 m and bench height of 35 m.

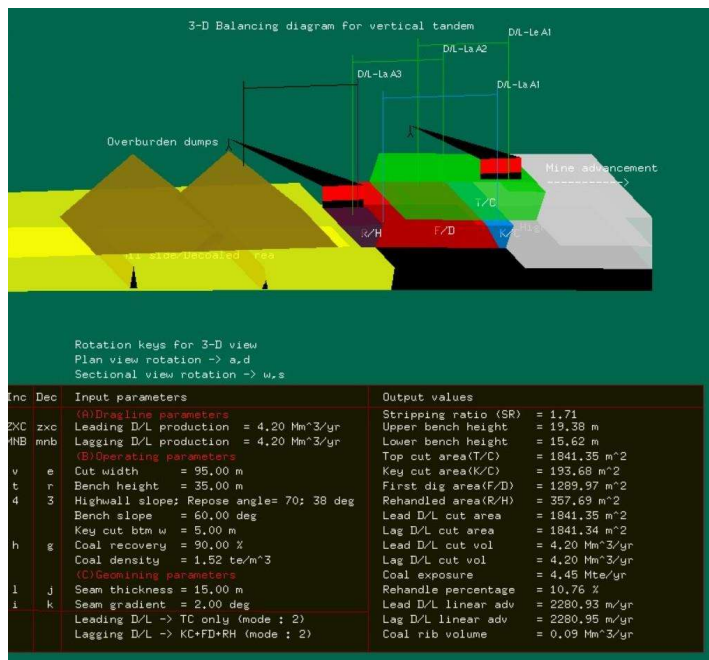


Figure 6.9.21: 3-dimensional balancing diagram for cut width of 95 m and bench height of 35 m.

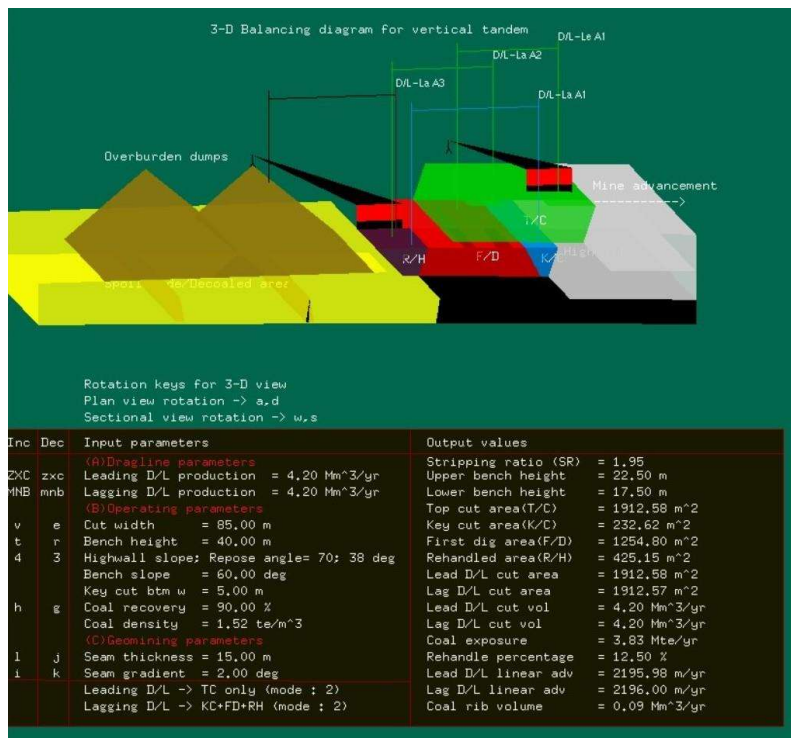


Figure 6.9.22: 3-dimensional balancing diagram for cut width of 85 m and bench height of 40 m.

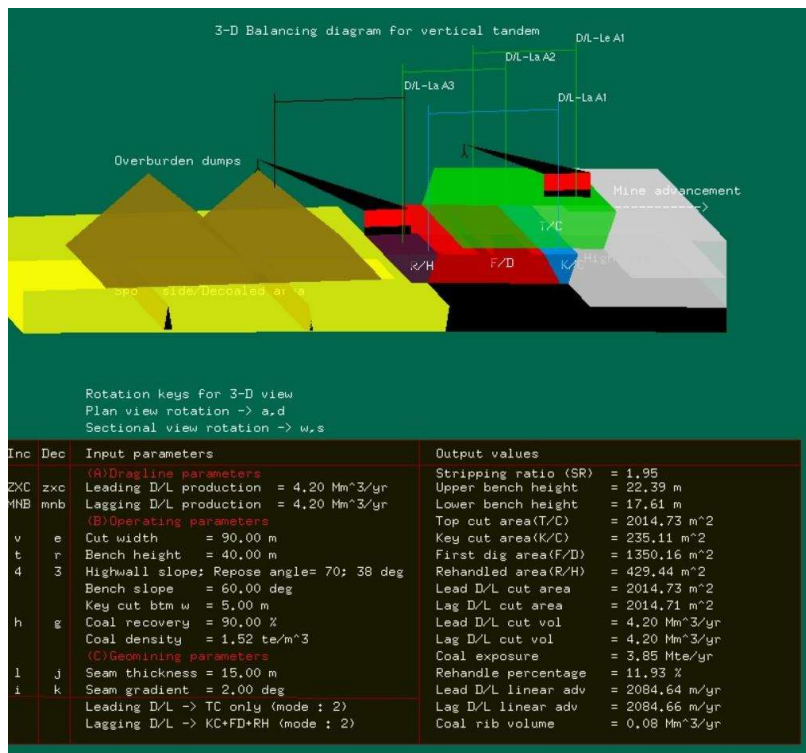


Figure 6.9.23: 3-dimensional balancing diagram for cut width of 90 m and bench height of 40 m.

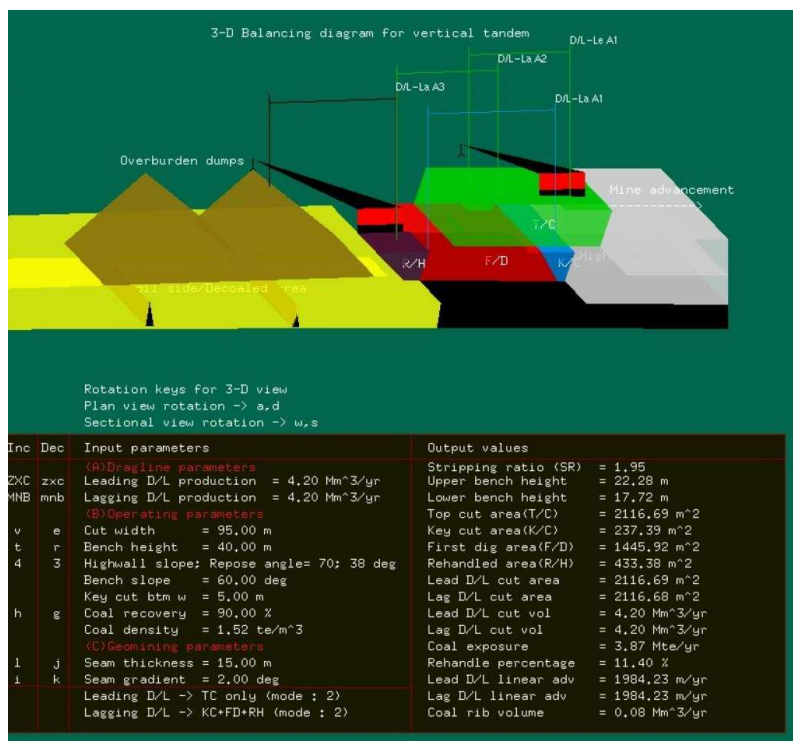


Figure 6.9.24: 3-dimensional balancing diagram for cut width of 95 m and bench height of 40 m.

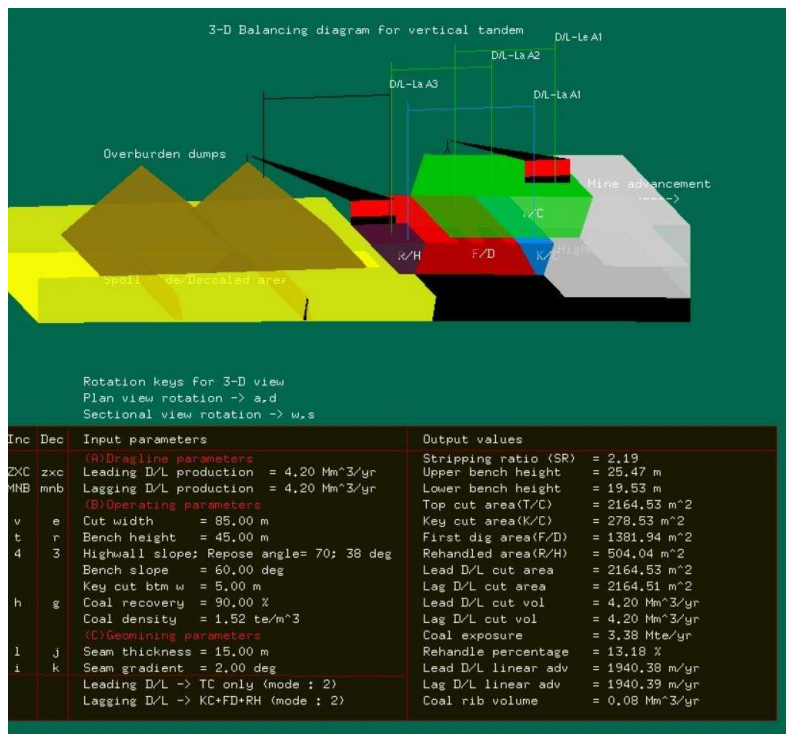


Figure 6.9.25: 3-dimensional balancing diagram for cut width of 85 m and bench height of 45 m.

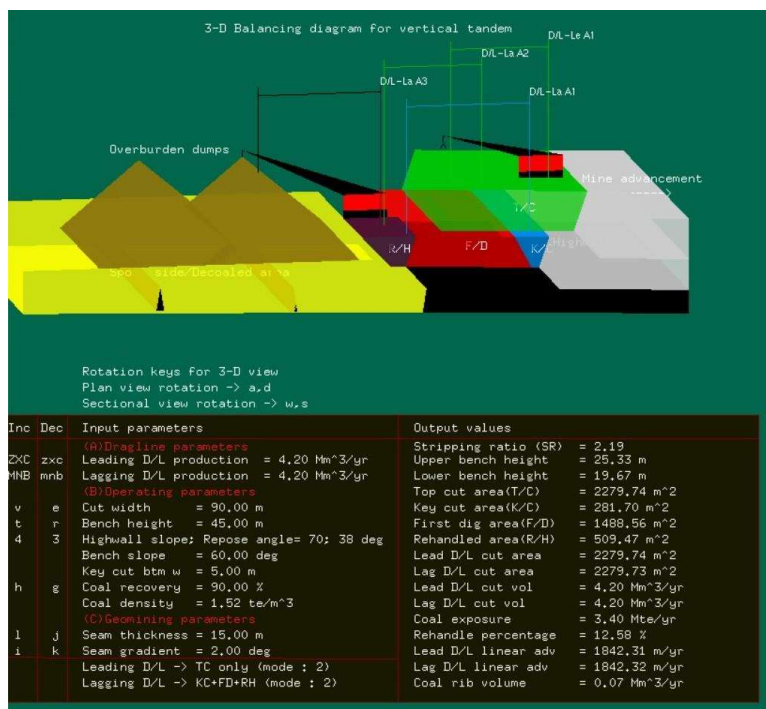


Figure 6.9.26: 3-dimensional balancing diagram for cut width of 90 m and bench height of 45 m.

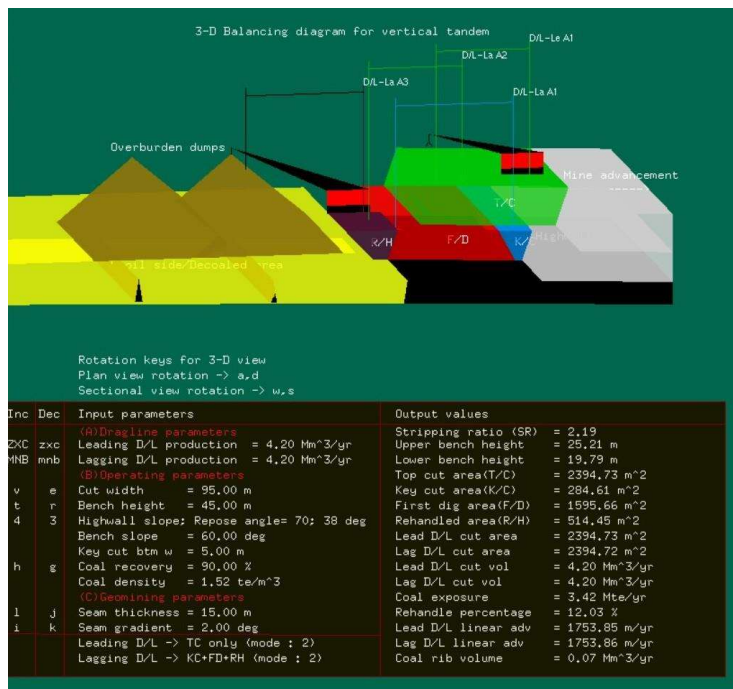


Figure 6.9.27: 3-dimensional balancing diagram for cut width of 95 m and bench height of 45 m.

APPENDIX-2 Computer program codes of three-dimensional balancing

diagram for horizontal tandem operation

```
#include<GL/gl.h>
#include<GL/glu.h>
#include<GL/glut.h>
#include<iostream>
#include<bits/stdc++.h>

#define PI 3.1416

using namespace std;

float anglex=0;

float angley=0;

float angle=35;

//st=seam thickness, bhl= lower bench height, bhu=upper bench height

//cw=cut width, kwt=key cut width top, kwb= key cut width bottom

//dragline specifications      //DL1= leading  DL2=lagging

int LeBucket= 24, LeBoom=96, LaBucket=24, LaBoom=96;

float coalRibVol,stripping;

string Le=to_string(LeBucket)+"/"+to_string(LeBoom);

string La=to_string(LaBucket)+"/"+to_string(LaBoom);

int maxreach=(LeBoom*sinf(0.40))-5;

// DL24/96 - 3.42/88R , DL15/90 - 2.44,83R , DL10/70 - 1.67, 66R  DL20/90 - 3.04(120 deg swing
angle) or 3.37(90 deg swing angle) ,83R

float DL1prod=3.42, reach1=83, DL2prod=3.42, reach2=88;

float dumph=41.5;

//geology

float st=18, dip=0, coal_dens=1.52;

//pit design

float cw=75, bht=35, bhl=bht, bhu=0;      //st- seam thickness, bhl=lower(main) bench height,
bhu- bench height upper

float castPercentage=0;

// kwt -key cut width top ,kwb- key cut width bottom

float riba=80,ha=70, ba=60, balg=45, aor=38;    //riba- angle of coal extraction, ha- highwall angle, ba-
bench angle, balg- bench angle for loose ground, aor- angle of repose
```



```

float recovery=0.9;

float rad= 3.14/180;


float keycut=0.0, firstdig=0.0, rehandle=0.0, topcut=0.0, firstcut=0.0,areaLead,areaLag;

float prod_ratio= DL1prod/DL2prod;

float x,y,z;

float a=0,b=0;

string s;

//float stripping_ratio=((keycut+firstcut+firstdig)*linear_DL1)/

float posx,posy,xpos;

string set2decimal(float fl){

    string str;

    std::stringstream stream;

    stream << std::fixed << std::setprecision(2) << fl;

    str = stream.str();

    return str;

}

float capacity(int bucket){

    float annualProduction;// variable name set to annual production from cap

    //swell=39

    //Cap= B/C*F*S*M*K*3600*8760 m3

    if(bucket==10) annualProduction=1.67;    //C=70

    else if(bucket==15) annualProduction=2.44; //72

    else if(bucket==24) annualProduction=3.42; //80

    return annualProduction;

}


int reach(int boom){

    int r;

    if(boom==96) r=88;

    else if(boom==90) r=83;

    else if(boom==70) r=66;

```

```

//  r=boom*cos(angle);

    return r;

}


void output(){
    x=-200;

    y=-80;

    z=0;

    glTranslated(0,a,0);

    glBegin(GL_QUADS);//Colour of the box
    glColor3f(0.1,0.1, 0);
    glVertex3f(x-60,y-170,z-10);
    glVertex3f(-x+50,y-170,z-10);
//  glColor3f(.35,.35,.67);
    glVertex3f(x-60,y+10,z-10);
    glEnd();

    glBegin(GL_LINES);//box lines
    glColor4f(0.5,0,0,1);
    glVertex3f(x-60,y+10,z);
    glVertex3f(-x+50,y+10,z);
    glVertex3f(x-60,y-170,z);
    glVertex3f(-x+50,y-170,z);
    glVertex3f(x-60,y-5,z);
    glVertex3f(-x+50,y-5,z);
    glVertex3f(x-60,y-170,z);
    glVertex3f(x-60,y+10,z)
    glVertex3f(x-30,y-170,z);
    glVertex3f(x-30,y+10,z);
    glVertex3f(x-10,y-170,z);
    glVertex3f(x-10,y+10,z);
    glVertex3f(-x+50,y-170,z);

```

```

glVertex3f(-x+50,y+10,z);

glVertex3f(x+200+10,y-170,z);

glVertex3f(x+200+10,y+10,z);

glEnd();


glColor3f(1,1,1);

s="Input Parameters";

glRasterPos3f(x,y,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

glColor3f(1, 1, 1);

s="Rotation keys for 3-D view";

glRasterPos3f(x,y+35,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Plan view rotation -> a,d";

glRasterPos3f(x,y+25,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Sectional view rotation -> w,s";

glRasterPos3f(x,y+15,z);

glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Inc";

glRasterPos3f(x-50,y,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

glColor3f(1, 1, 1);

s="Dec";

glRasterPos3f(x-28,y,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

```

```

    glColor3f(0.9, 0.1, 0.1);
//   s="Lead DL      = " + Le ;
//   glRasterPos3f(x,y-20,z);
//   for(int i=0;i<s.length();i++)
//       glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);//will see in future if we want to add

//   s="Lag DL      = " + La ;
//   glRasterPos3f(x,y-30,z);
//   for(int i=0;i<s.length();i++)
//       glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
    glColor3f(0.9, 0.1, 0.1);
    s="(A)Geomining parameters";
    glRasterPos3f(x,y-20,z);
    for(int i=0;i<s.length();i++)
        glColor3f(1,1,1);
    s="Cut width      = " + set2decimal(cw) + " m";
    glRasterPos3f(x,y-80,z);
    for(int i=0;i<s.length();i++)
        glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
    s="j";
    glRasterPos3f(x-20,y-80,z);
    for(int i=0;i<s.length();i++)
        glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
    s="l";
    glRasterPos3f(x-50,y-80,z);
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
    s="Bench height  = " + set2decimal(bhl) + " m";
    glRasterPos3f(x,y-90,z);
    for(int i=0;i<s.length();i++)

        glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
    s="i";

```

```

glRasterPos3f(x-50,y-90,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="k";


glRasterPos3f(x-20,y-90,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);


s="Highwall slope = " + set2decimal(ha) + " deg";

glRasterPos3f(x,y-60,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="3";

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="4";

glRasterPos3f(x-50,y-60,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Bench slope  = " + set2decimal(ba) + " deg";

glRasterPos3f(x,y-70,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

// s="5";

// glRasterPos3f(x-20,y-70,z);

// for(int i=0;i<s.length();i++)

//     glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

// s="6";

// glRasterPos3f(x-50,y-70,z);

// for(int i=0;i<s.length();i++)

//     glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

```



```

s="Seam thickness = " + set2decimal(st) + " m";

glRasterPos3f(x,y-30,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="t";

glRasterPos3f(x-50,y-30,z);

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);


s="r";

glRasterPos3f(x-20,y-30,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Seam gradient = " + set2decimal(dip) + " deg";

glRasterPos3f(x,y-40,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="2";

glRasterPos3f(x-50,y-40,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="1";

glRasterPos3f(x-20,y-40,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

glColor3f(0.9, 0.1, 0.1);

s="(B)Operating parameters";

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="(C)Dragline parameters";

glRasterPos3f(x,y-140,z);

```

```

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

glColor3f(1,1,1);


s="Coal density  = " + set2decimal(coal_dens) + " te/m^3";

glRasterPos3f(x,y-100,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);


s="Key cut btm w = " + set2decimal(kwb) + " m";

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);


s="Coal recovery  = " + set2decimal(recovery*100) + " %";

glRasterPos3f(x,y-120,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Angle of repose  = 38 deg";

glRasterPos3f(x,y-130,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);


s="h";

glRasterPos3f(x-50,y-120,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);


s="g";

glRasterPos3f(x-20,y-120,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

```

```

s="Leading D/L production = "+ set2decimal(DL1prod)+ " Mm^3/yr";

glRasterPos3f(x,y-150,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);


s="ZXC";

glRasterPos3f(x-55,y-150,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);


s="zxc";

glRasterPos3f(x-28,y-150,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);


s="Lagging D/L production = "+set2decimal(DL2prod)+ " Mm^3/yr";

glRasterPos3f(x,y-160,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);


s="MNB";

glRasterPos3f(x-55,y-160,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);


s="mnb";

glRasterPos3f(x-28,y-160,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);


x=20;

```

```

glColor3f(1, 1, 1);
s="Output Values";
glRasterPos3f(x,y,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
glColor3f(1,1,1);

s="Key cut area (K/C) = " + set2decimal(keycut) + " m^2";
glRasterPos3f(x,-100,0);
for(int i=0;i<s.length();i++)
s="Stripping ratio (SR) = " + set2decimal(stripping);
glRasterPos3f(x,-92,0);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="First cut area (F/C) = " + set2decimal(firstcut) + " m^2";
glRasterPos3f(x,-110,0);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="First dig area (F/D) = " + set2decimal(firstdig) + " m^2";
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Rehandled area (R/H) = " + set2decimal(rehandle) + " m^2";
glRasterPos3f(x,-130,0);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Leading D/L cut area = " + set2decimal(areaLead) + " m^2";

```

```

glRasterPos3f(x,-140,0);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);


s="Lagging D/L cut area  = " + set2decimal(areaLag) + " m^2";

glRasterPos3f(x,-150,0);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);


s="Lead D/L cut vol    = " + set2decimal(areaLead*linear_DL1/1000000) + " Mm^3/yr";

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);


s="Lag D/L cut vol    = " + set2decimal(areaLag*linear_DL2/1000000) + " Mm^3/yr";

glRasterPos3f(x,-170,0);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);


s="Coal exposure      = " + set2decimal(coalExpDL1) + " Mte/yr";

glRasterPos3f(x,-180,0);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Rehandle percentage  = " + set2decimal(rh_per) + " %";

glRasterPos3f(x,-190,0);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Lead D/L linear adv  = " + set2decimal(linear_DL1) + " m/yr";

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Lag D/L linear adv   = " + set2decimal(linear_DL2) + " m/yr";

```

```

glRasterPos3f(x,-210,0);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Max dump height      = " + set2decimal(maxreach)+ " m";

glRasterPos3f(x,-230,0);

for(int i=0;i<s.length();i++)

s="Coal rib volume      = " + set2decimal(coalRibVol/1000000)+ " Mm^3/yr"+"rib";

glRasterPos3f(x,-240,0);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

}

void display(){

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);//To set to previous values.

    glLoadIdentity();//To set the matrix to identity.

    float x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16,x17,x18,dx;

    float y1,y2,y3,y4,y5,y6,y7,y8,y9,y10,y11,y12,y13,y14,y15,y16,y17,y18,dy;

    float ribx, riby;

    float xd[15],yd[15];

    bhl=bht;

    // DL1prod=capacity(LeBucket);

    // DL2prod=capacity(LaBucket);

    reach1=reach(LeBoom);

    reach2=reach(LaBoom);

    prod_ratio= DL1prod/DL2prod;

    x1=0;                                //A

    x2=cw;                                //B

    x3=x2+cw;                             //B'

    y1=0;

    y2=-x2*tan(dip*rad);

    y3=-x3*tan(dip*rad);

```


$dx = st \cdot \cos(dip \cdot rad) / \sin((dip + riba) \cdot rad) \cdot \cos(riba \cdot rad);$

$dy = st \cdot \cos(dip \cdot rad) / \sin((dip + riba) \cdot rad) \cdot \sin(riba \cdot rad);$

$x4 = x1 + dx; \quad //C$

$x5 = x2 + dx; \quad //D$

$x6 = x3; \quad //D'$

$y4 = y1 + dy;$

$y5 = y2 + dy;$

$y6 = y3 + st;$

$x7 = x4 + dx; \quad //E$

$x8 = x5 + dx; \quad //F$

$x9 = x6; \quad //F'$

$y7 = y4 + dy;$

$y8 = y5 + dy;$

$y9 = y6 + bhl;$

$x10 = x7 + bermlb; // * \cos(dip \cdot rad);$

$y10 = y7 - bermlb \cdot \tan(dip \cdot rad);$

$x11 = x10 + cw;$

$y11 = y10 - cw \cdot \tan(dip \cdot rad);$

$dx = bhu \cdot \cos(dip \cdot rad) / \sin((dip + ha) \cdot rad) \cdot \cos(ha \cdot rad);$

$dy = bhu \cdot \cos(dip \cdot rad) / \sin((dip + ha) \cdot rad) \cdot \sin(ha \cdot rad);$

$x13 = x10 + dx;$

$y13 = y10 + dy;$

$x14 = x11 + dx;$

$y14 = y11 + dy;$

$x12 = x9;$

$y12 = y9 + bhu$

$dx = bhl \cdot \cos(dip \cdot rad) / \sin((dip + (180 - ba)) \cdot rad) \cdot \cos((180 - ba) \cdot rad);$

$dy = bhl \cdot \cos(dip \cdot rad) / \sin((dip + (180 - ba)) \cdot rad) \cdot \sin((180 - ba) \cdot rad);$

$x15 = x5 - kwb; \quad //H$

```

//x16=x8-kwt; //J
x16=x15+dx;
y15=y5+kwb*tan(dip*rad);
//y16=y8+kwt*tan(dip*rad);
y16=y15+dy;
posx=(x16-x7)/2;
posy=posx*tan(dip*rad);
x17=x15;//-posx; //G
x18=x16;//-posx; //I
y17=y15;//+posy;
y18=y16;//+posy;
//dump side

xd[0]=x1-cw; //T
yd[0]=-xd[0]*tan(dip*rad);
xd[1]=x1-2*cw; //T'
yd[1]=-xd[1]*tan(dip*rad);
xd[2]=xd[1]; //T''
yd[2]=y7;
xd[3]=(x8+x16)/2 - reach1; //L
yd[3]=y7;
xd[5]=x4-bermrh; //O
yd[5]=y4+bermrh*tan(dip*rad);
xd[6]=(x16+x18)/2 - reach1; //N
yd[6]=y7;
xd[7]=((tan(aor*rad)*xd[6] - tan((180-dip)*rad)*x4)-(yd[6]-y4))/(tan(aor*rad)-tan((180-dip)*rad));
//M
yd[7]=xd[7]*tan(aor*rad)+yd[6]-tan(aor*rad)*xd[6];
xd[8]=(y7-(yd[5]-tan((180-balg)*rad)*xd[5]))/tan((180-balg)*rad); //P
yd[8]=y7;
xd[9]=xd[8]-bermrh-5; //S
yd[9]=y7;

```

```

yd[10]=y7+dumph; //Q
xd[10]=xd[9]- dumph/tan(rad*aor);
xd[11]= xd[10]-(xd[9]-xd[10]); //R
yd[11]= y7;
ribx=2*st/tan(riba*rad);
riby= ribx*tan(dip*rad);
//optimization

posx=0;
//float a,b,c,x,y,z;
keycut=0.5*((x5-x15)+(x8-x16))*bhl;// cos(dip*rad) ;
firstcut=0.5*((x15-x17)+(x16-x18))*bhl;//cos(dip*rad);
firstdig=0.5*((x17-x4)+(x18-x7))*bhl;//cos(dip*rad);
castPercentage=((57.5*bht)/cw)+18);
y= x*tan(aor*rad)+yd[6]-tan(aor*rad)*xd[6];
a=0.5*((x7-xd[6])+(x4-x))*(y7-y4);
b=0.5*(xd[5]-x)*((xd[5]-x)/2)*tan(aor*rad);
c=0.5*((x4-xd[5])+(x7-xd[8]))*(y7-y4);*/
//rehandle=a-b;
rehandle= 0.5*(bermrh+ (x7-xd[8]))*bhl - 0.5*bermrh*bermrh*tan(dip*rad);
rehandle=1.1*rehandle;
areaLead=keycut+firstcut;
areaLag=firstdig+rehandle;
if(areaLag >= (areaLead/prod_ratio)){
    while(areaLag >= (areaLead/prod_ratio)){
        posx+=0.00001;
        posy=posx*tan(dip*rad);
        x17=x15-posx;
        x18=x16-posx;
        y17=y15+posy;
        y18=y16+posy;
    }
}

```

```

        firstcut=0.5*((x15-x17)+(x16-x18))*bhl;//cos(dip*rad);
        firstdig=0.5*((x17-x4)+(x18-x7))*bhl;//cos(dip*rad);
        areaLead=keycut+firstcut;
        areaLag=firstdig+rehandle;
    }
}

//optimization ends

prodKC= keycut/areaLead*DL1prod;
prodFC= firstcut/areaLead*DL1prod;
prodFD= firstdig/areaLag*DL2prod;
prodRH= rehandle/areaLag*DL2prod;
linear_DL1= DL1prod*1000000/areaLead;
linear_DL2= DL2prod*1000000/areaLag;
coalExpDL2= linear_DL2*cw*st*coal_dens*recovery/1000000;
coalExp= coalExpDL1+coalExpDL2;
rh_per= rehandle*100/(keycut+firstcut+firstdig);
coalRibVol=((ribx*st)/2)*linear_DL1;
stripping=((keycut+firstdig+firstcut)*linear_DL1)/(coalExpDL1*1000000);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);//specify pixel arithmetic
glEnable( GL_BLEND );// To enable blending

//display the values

/*string str;

std::stringstream stream;

stream << std::fixed << std::setprecision(2) << keycut;

str = stream.str();*/

output();

glTranslated(0,10,0);

glRotatef(anglex,1.0,0.0,0.0);

glRotatef(angley,0.0,1.0,0.0);

//draw

```

```

glColor3f(1,1,1);
s= "R/H";
glRasterPos3f((xd[8]+x7)/2,(y4+y7)/2,70);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
z=-200;
glBegin(GL_QUADS);                                //rehandle

```

```

glColor4f(0.33,0.06,0.18,0.8);
glVertex3f(x4,y4,70);
glVertex3f(xd[8],yd[8],70);
glVertex3f(xd[5],yd[5],70);
glVertex3f(x4,y4,0);
glVertex3f(x7,y7,0);
glVertex3f(xd[8],yd[8],0);
glVertex3f(xd[5],yd[5],0);
glVertex3f(x4,y4,70);
glVertex3f(x7,y7,70);
glVertex3f(x7,y7,00);
glVertex3f(x4,y4,00);
glVertex3f(x7,y7,70);
glVertex3f(xd[8],yd[8],70);
glVertex3f(xd[8],yd[8],00);
glVertex3f(x7,y7,00);
glVertex3f(xd[8],yd[8],70);
glVertex3f(xd[5],yd[5],00);
glVertex3f(xd[8],yd[8],00);
glVertex3f(xd[5],yd[5],70);
glVertex3f(x4,y4,70);
glVertex3f(x4,y4,00);
glVertex3f(xd[5],yd[5],00);

```

```

glEnd();

glBegin(GL_TRIANGLES);

//coal rib
glColor3f(0,0,0);
glVertex3f(xd[0],yd[0],150);
glVertex3f(xd[0]+ribx, yd[0]-riby,150);
glVertex3f(xd[0]+ribx/2, yd[0]-riby/2+st,150);
glEnd();

glBegin(GL_QUADS);
glVertex3f(xd[0],yd[0],150);
glVertex3f(xd[0]+ribx, yd[0]-riby,150);
glVertex3f(xd[0]+ribx, yd[0]-riby,z);
glVertex3f(xd[0],yd[0],z);
glVertex3f(xd[0]+ribx, yd[0]-riby,150);
glVertex3f(xd[0]+ribx/2, yd[0]-riby/2+st,150);
glVertex3f(xd[0],yd[0],150);
glVertex3f(xd[0]+ribx/2, yd[0]-riby/2+st,150);
glVertex3f(xd[0]+ribx/2, yd[0]-riby/2+st,z);
glVertex3f(xd[0],yd[0],z);
glEnd();

glBegin(GL_TRIANGLES);                                //coal rib 2
glColor3f(0,0,0);
glVertex3f(xd[1],yd[1],150);
glVertex3f(xd[1]+ribx, yd[1]-riby,150);
glVertex3f(xd[1]+ribx/2, yd[1]-riby/2+st,150);
glEnd();

glBegin(GL_QUADS);
glVertex3f(xd[1],yd[1],150);
glVertex3f(xd[1]+ribx, yd[1]-riby,150);
glVertex3f(xd[1]+ribx, yd[1]-riby,z);
glVertex3f(xd[1],yd[1],z);

```



```

glVertex3f(xd[1]+ribx, yd[1]-riby,150);
glVertex3f(xd[1]+ribx/2, yd[1]-riby/2+st,150);
glVertex3f(xd[1]+ribx/2, yd[1]-riby/2+st,z);
glVertex3f(xd[1]+ribx, yd[1]-riby,z);
glVertex3f(xd[1],yd[1],150);
glVertex3f(xd[1]+ribx/2, yd[1]-riby/2+st,150);
glVertex3f(xd[1]+ribx/2, yd[1]-riby/2+st,z);
glVertex3f(xd[1],yd[1],z);
glEnd();
x=-2*cw-20;
if(x> xd[11]-10) x=xd[11]-10;
glColor3f(1, 1, 1);
s= "Spoil side/Decoaled area";
glRasterPos3f((x16+x8)/2-2+20-300,(y15+y16)/2-10,50);
for(int i=0;i<s.length();i++)
s= "Overburden dumps";
glRasterPos3f((x16+x8)/2-2+20-300,(y15+y16)/2+75,50);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);


glBegin(GL_POLYGON);                //dump
glColor4f(1.0,1.0,0.0,0.8);
glVertex3f(x1,y1,150);
glVertex3f(x-70,-x*tan(rad*dip),150);
glVertex3f(x-70,yd[2],150);
glVertex3f(xd[8],yd[8],150);
glVertex3f(xd[5],yd[5],150);
glVertex3f(x4,y4,150);
glEnd();
glBegin(GL_QUADS);

```

```

glVertex3f(x1,y1,150);
glVertex3f(x-70,-x*tan(rad*dip),150);
glVertex3f(x-70,-x*tan(rad*dip),z);
glVertex3f(x1,y1,z);

glVertex3f(x-70,-x*tan(rad*dip),150);
glVertex3f(x-70,yd[2],150);
glVertex3f(x-70,yd[2],z);
glVertex3f(x-70,-x*tan(rad*dip),z);

glVertex3f(x-70,yd[2],150);
glVertex3f(xd[8],yd[8],150);
glVertex3f(xd[8],yd[8],z);
glVertex3f(x-70,yd[2],z);
glVertex3f(xd[8],yd[8],150);
glVertex3f(xd[5],yd[5],150);
glVertex3f(xd[8],yd[8],z);
glVertex3f(xd[5],yd[5],150);
glVertex3f(x4,y4,150);
glVertex3f(x4,y4,z);
glVertex3f(xd[5],yd[5],z);
glEnd();
glBegin(GL_POLYGON);
glColor4f(1.0,1.0,0.0,0.8);
glVertex3f(x1,y1,z);
glVertex3f(x,-x*tan(rad*dip),z);
glVertex3f(x,yd[2],z);
glVertex3f(xd[8],yd[8],z);
glVertex3f(xd[5],yd[5],z);
glVertex3f(x4,y4,z);
glEnd();

```

```

glBegin(GL_TRIANGLES);           //dump triangle
glColor4f(0.55,0.46,0.05,0.95);
glVertex3f(xd[9],yd[9],150);
glVertex3f(xd[10],yd[10],150);
glVertex3f(xd[11],yd[11],150);
glEnd();

```

```

glBegin(GL_TRIANGLES);           //dump triangle 2 by vaibhav
glColor4f(0.55,0.46,0.05,0.95);
glVertex3f(xd[9]-70,yd[9],150);
glVertex3f(xd[10]-70,yd[10],150);
glVertex3f(xd[11]-70,yd[11],150);
glEnd();

```

```

glBegin(GL_QUADS);
glVertex3f(xd[9],yd[9],150);
glVertex3f(xd[10],yd[10],150);
glVertex3f(xd[10],yd[10],0);
glVertex3f(xd[9],yd[9],0);
glVertex3f(xd[11],yd[11],0);
glVertex3f(xd[10],yd[10],0);
glVertex3f(xd[9],yd[9],150);
glVertex3f(xd[11],yd[11],150);
glVertex3f(xd[11],yd[11],0);
glVertex3f(xd[9],yd[9],0);
glEnd();

```

```

glBegin(GL_QUADS); // dump triangle 2 3d by vaibhav
glVertex3f(xd[9]-70,yd[9],150);
glVertex3f(xd[10]-70,yd[10],150);
glVertex3f(xd[10]-70,yd[10],0);
glVertex3f(xd[9]-70,yd[9],0);

```

```

glVertex3f(xd[10]-70,yd[10],150);
glVertex3f(xd[11]-70,yd[11],150);
glVertex3f(xd[11]-70,yd[11],0);
glVertex3f(xd[10]-70,yd[10],0);

glVertex3f(xd[9]-70,yd[9],150);
glVertex3f(xd[11]-70,yd[11],150);
glVertex3f(xd[11]-70,yd[11],0);
glVertex3f(xd[9]-70,yd[9],0);
glEnd();

glBegin(GL_TRIANGLES);
glColor4f(0.55,0.46,0.0,0.95);
glVertex3f(xd[9],yd[9],00);
glVertex3f(xd[10],yd[10],00);
glVertex3f(xd[11],yd[11],00);
glEnd();

glBegin(GL_TRIANGLES);
glColor4f(0.55,0.46,0.05,0.95);
glVertex3f(xd[9],yd[9],150);
glVertex3f(xd[10],yd[10],150);
glEnd();

glBegin(GL_QUADS);
glVertex3f(xd[9],yd[9],150);
glVertex3f(xd[10],yd[10],150);
glVertex3f(xd[10],yd[10],0);
glVertex3f(xd[9],yd[9],0);
glVertex3f(xd[10],yd[10],150);
glVertex3f(xd[11],yd[11],150);
glVertex3f(xd[11],yd[11],0);
glVertex3f(xd[10],yd[10],0);
glVertex3f(xd[9],yd[9],150);

```

```

glVertex3f(xd[11],yd[11],150);
glVertex3f(xd[11],yd[11],0);
glVertex3f(xd[9],yd[9],0);
glEnd();
glBegin(GL_TRIANGLES);
glColor4f(0.55,0.46,0.0,0.95);
glVertex3f(xd[9],yd[9],00);
glVertex3f(xd[10],yd[10],00);
glVertex3f(xd[11],yd[11],00);
glEnd();
glColor3f(1, 1, 1);
//bench side

```

```

glBegin(GL_QUADS);           //coal seam part1
glColor4f(0.0,0.0,0.0,1);
glVertex3f(x1,y1,150);
glVertex3f(x2,y2,150);
glVertex3f(x5,y5,150);
glVertex3f(x4,y4,150);
glVertex3f(x1,y1,z);
glVertex3f(x2,y2,z);
glVertex3f(x4,y4,z);
glVertex3f(x1,y1,150);
glVertex3f(x4,y4,150);
glVertex3f(x4,y4,z);
glVertex3f(x1,y1,z);
glVertex3f(x2,y2,150);
glVertex3f(x5,y5,150);
glVertex3f(x5,y5,z);
glVertex3f(x2,y2,z);
glVertex3f(x1,y1,150);
glVertex3f(x2,y2,150);

```

```

glVertex3f(x2,y2,z);
glVertex3f(x1,y1,z);
glVertex3f(x4,y4,150);
glVertex3f(x5,y5,150);
glVertex3f(x5,y5,z);
glVertex3f(x4,y4,z);

glEnd();

```

```

glBegin(GL_QUADS);                                //coal seam part 2
glColor4f(0.0,0.0,0.0,1);
glVertex3f(x2,y2,150);
glVertex3f(x3,y3,150);
glVertex3f(x6,y6,150);
glVertex3f(x5,y5,150);
glVertex3f(x2,y2,z);
glVertex3f(x3,y3,z);
glVertex3f(x6,y6,z);
glVertex3f(x5,y5,z);
glVertex3f(x2,y2,150);
glVertex3f(x5,y5,150);
glVertex3f(x5,y5,z);
glVertex3f(x2,y2,z);
glVertex3f(x3,y3,150);
glVertex3f(x6,y6,150);
glVertex3f(x3,y3,z);
glVertex3f(x2,y2,150);
glVertex3f(x3,y3,150);
glVertex3f(x3,y3,z);
glVertex3f(x2,y2,z);
glVertex3f(x5,y5,150);

```

```

glVertex3f(x6,y6,150);
glVertex3f(x6,y6,z);
glVertex3f(x5,y5,z);
glEnd();
glColor3f(1,1,1);
s= "F/D";
glRasterPos3f((x4+x18)/2,(y4+y18)/2,70);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

```

```

glBegin(GL_QUADS);                                //first dig
glColor4f(1.0,0.0,0.0,0.7);
glVertex3f(x4,y4,70);
glVertex3f(x17,y17,70);
glVertex3f(x18,y18,70);
glVertex3f(x7,y7,70);
glVertex3f(x4,y4,z);
glVertex3f(x17,y17,z);
glVertex3f(x18,y18,z);
glVertex3f(x7,y7,z);
glVertex3f(x7,y7,70);
glVertex3f(x7,y7,z);
glVertex3f(x4,y4,z);
glVertex3f(x17,y17,70);
glVertex3f(x18,y18,70);
glVertex3f(x18,y18,z);
glVertex3f(x17,y17,z);
glVertex3f(x4,y4,70);
glVertex3f(x17,y17,70);
glVertex3f(x17,y17,z);
glVertex3f(x7,y7,70);

```



```

glVertex3f(x18,y18,70);
glVertex3f(x18,y18,z);
glVertex3f(x7,y7,z);
glEnd();

glColor3f(1,1,1);
s= "F/C";
glRasterPos3f((x18+x16)/2,(y15+y16)/2,50);
for(int i=0;i<s.length();i++)

glBegin(GL_QUADS);                                //first cut
glColor4f(0.5,1.0,0.5,0.7);
glVertex3f(x17,y17,50);
glVertex3f(x15,y15,50);
glVertex3f(x16,y16,50);
glVertex3f(x18,y18,50);
glVertex3f(x17,y17,z);
glVertex3f(x15,y15,z);
glVertex3f(x16,y16,z);
glVertex3f(x18,y18,z);
glVertex3f(x17,y17,50);
glVertex3f(x18,y18,50);
glVertex3f(x18,y18,z);
glVertex3f(x17,y17,z);
glVertex3f(x15,y15,50);
glVertex3f(x16,y16,50);
glVertex3f(x16,y16,z);
glVertex3f(x15,y15,z);
glVertex3f(x17,y17,50);
glVertex3f(x15,y15,50);
glVertex3f(x15,y15,z);
glVertex3f(x17,y17,z);

```

```

glVertex3f(x18,y18,50);
glVertex3f(x16,y16,50);
glVertex3f(x16,y16,z);
glVertex3f(x18,y18,z);
glEnd();
glColor3f(1,1,1);
s= "K/C";
glRasterPos3f((x16+x8)/2-2,(y15+y16)/2,50);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s= "Highwall";
glRasterPos3f((x16+x8)/2-2+20,(y15+y16)/2,50);
for(int i=0;i<s.length();i++)
glColor3f(1,1,1);
s= "Mine advancement";
glRasterPos3f((x16+x8)/2-2+20,(y15+y16)/2+40,50);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s= "----->";
glRasterPos3f((x16+x8)/2-2+20,(y15+y16)/2+30,50);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s= "3-D Balancing diagram for horizontal tandem";
glRasterPos3f(x,y+255,50);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

glBegin(GL_QUADS);                                //key cut
glColor4f(0.0,0.6,1.0,0.7);
glVertex3f(x15,y15,50);
glVertex3f(x5,y5,50);

```

```

glVertex3f(x8,y8,50);
glVertex3f(x16,y16,50);

glVertex3f(x15,y15,z);
glVertex3f(x5,y5,z);
glVertex3f(x8,y8,z);
glVertex3f(x16,y16,z);
glVertex3f(x15,y15,50);
glVertex3f(x16,y16,50);
glVertex3f(x16,y16,z);
glVertex3f(x15,y15,z);
glVertex3f(x5,y5,50);
glVertex3f(x8,y8,50);
glVertex3f(x8,y8,z);
glVertex3f(x5,y5,z);

glVertex3f(x15,y15,50);
glVertex3f(x5,y5,50);
glVertex3f(x5,y5,z);
glVertex3f(x15,y15,z);
glVertex3f(x16,y16,50);
glVertex3f(x8,y8,50);
glVertex3f(x8,y8,z);
glVertex3f(x16,y16,z);
glEnd();

glBegin(GL_QUADS);           //lower bench part 3
glColor4f(0.8,0.8,0.8,0.9);
glVertex3f(x5,y5,150);
glVertex3f(x6,y6,150);
glVertex3f(x9,y9,150);
glVertex3f(x8,y8,150)
glVertex3f(x5,y5,z);

```

```

glVertex3f(x6,y6,z);
glVertex3f(x9,y9,z);
glVertex3f(x8,y8,z);
glVertex3f(x5,y5,150);
glVertex3f(x8,y8,150);
glVertex3f(x8,y8,z);
glVertex3f(x5,y5,z)
glVertex3f(x6,y6,150);
glVertex3f(x9,y9,150);
glVertex3f(x9,y9,z);
glVertex3f(x6,y6,z);
glVertex3f(x5,y5,150);
glVertex3f(x6,y6,150);
glVertex3f(x6,y6,z);
glVertex3f(x5,y5,z);
glVertex3f(x8,y8,150);
glVertex3f(x9,y9,150);
glVertex3f(x9,y9,z);
glVertex3f(x8,y8,z);
glEnd();

//axes
xpos=(x8+x16)/2;

glBegin(GL_LINES);           //DL1 Axis1 (keycut)
glColor4f(0.5,1.0,0.5,0.7);
glVertex3f(xpos,150,30);
glVertex3f(xpos,0,30);
glEnd();

glBegin(GL_LINES);
//glColor3f(0,0,0);
glVertex3f(xpos-reach1,150,30);
glVertex3f(xpos-reach1,0,30);
glEnd();

```

```

glBegin(GL_LINES);
//glColor3f(0,0,0);
glVertex3f(xpos,145,30);
glVertex3f(xpos-reach1,145,30);
glEnd();

glBegin(GL_QUADS);
    glColor3d(0,0,0);
    glVertex3f(xpos-15,y14,0);
    glVertex3f(xpos-15,y14+5,0);
    glVertex3f(xpos+15,y14+5,0);
    glVertex3f(xpos+15,y14,0);
    glEnd();

glBegin(GL_QUADS);
    glColor3d(1,0,0);
    glVertex3f(xpos-15,y14+15,0);
    glVertex3f(xpos-15,y14+5,0);
    glVertex3f(xpos+15,y14+5,0);
    glVertex3f(xpos+15,y14+15,0);
    glEnd();

glBegin(GL_TRIANGLES);
    glColor3f(0,0,0);
    glVertex3f(xpos-15,y14+15,0);
    glVertex3f(xpos+15,y14+15,0);
    glVertex3f(xpos-reach1,95,0);
    glEnd();

//here is dg code above
glBegin(GL_LINES);
//glColor3f(0,0,0);
glVertex3f(xpos-reach1,95,0);
glVertex3f(xpos-reach1,90,0);
glEnd();
glBegin(GL_LINES);

```

```

//glColor3f(0,0,0);

glVertex3f(xpos-reach1-2,87,0);
glVertex3f(xpos-reach1,90,0);

glEnd();

glBegin(GL_LINES);
//glColor3f(0,0,0);

glVertex3f(xpos-reach1+2,87,0);
glVertex3f(xpos-reach1,90,0);

glEnd();


glColor3f(1,1,1);
s="D/L-Le A1";

glRasterPos3f(xpos,150,30);
for(int i=0;i<s.length();i++)
xpos=(x18+x16)/2;

glLineWidth(3);

glBegin(GL_LINES);//DL1 Axis2
glColor4f(0.5,1.0,0.5,0.7);
glVertex3f(xpos,150,30);
glVertex3f(xpos,st,30);

glEnd();

glBegin(GL_LINES);
//glColor3f(0,0,0);

glVertex3f(xpos-reach1,150,30);
glVertex3f(xpos-reach1,st,30);

glEnd();

glBegin(GL_LINES);
//glColor3f(0,0,0);

glVertex3f(xpos,145,30);
glVertex3f(xpos-reach1,145,30);

glEnd();

glLineWidth(1);

```

```

s="D/L-Le A2";

glColor3f(1,1,1);

glRasterPos3f(xpos,120,30);

for(int i=0;i<s.length();i++)
{
    glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_10,s[i]);}

glBegin(GL_LINES);           //DL2 Axis1

glColor4f(0.33,0.06,0.2,0.8);

glVertex3f(xd[10]+reach2,130,60);

glVertex3f(xd[10]+reach2,y7,60);

glEnd();

glBegin(GL_LINES);

//glColor3f(0,0,0);

glVertex3f(xd[10],130,60);

glVertex3f(xd[10],y7,60);

glEnd();

glBegin(GL_LINES);

//glColor3f(0,0,0);

glVertex3f(xd[10],125,60);

glEnd();

glBegin(GL_QUADS);

glColor3d(0,0,0);

glVertex3f(xd[10]+reach2-15,y7,0);

glVertex3f(xd[10]+reach2-15,y7+5,0);

glVertex3f(xd[10]+reach2+15,y7+5,0);

glVertex3f(xd[10]+reach2+15,y7,0);

glEnd();

glBegin(GL_QUADS);

glColor3d(1,0,0);

glVertex3f(xd[10]+reach2-15,y7+15,0);

glVertex3f(xd[10]+reach2-15,y7+5,0);

glVertex3f(xd[10]+reach2+15,y7+5,0);

```



```

glVertex3f(xd[10]+reach2+15,y7+15,0);

    glEnd();
glBegin(GL_TRIANGLES);

    glColor3f(0,0,0);

glVertex3f(xd[10]+reach2-15,y7+15,0);
glVertex3f(xd[10]+reach2+15,y7+15,0);
glVertex3f(xd[10],100,0);

    glEnd();

//here is dg code above

glBegin(GL_LINES);

//glColor3f(0,0,0);

glVertex3f(xd[10],100,0);
glVertex3f(xd[10],90,0);

glEnd();

glBegin(GL_LINES);

//glColor3f(0,0,0);

glVertex3f(xd[10],90,0);
glVertex3f(xd[10]-2,85,0);

glEnd();

glBegin(GL_LINES);

//glColor3f(0,0,0);

glVertex3f(xd[10],90,0);
glVertex3f(xd[10]+2,85,0);

glEnd();

glColor3f(1,1,1);

s="D/L-La A1";

glRasterPos3f(xd[10]+reach2,130,60);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_10,s[i]);

//glFlush();

glutSwapBuffers();    //to swap the front and back buffers for animation purpose
}

```

```

void reshape(int h, int w){

    //viewport- the rectangular area inside the window where the graphics are displayed

    glViewport(0, 0, (GLsizei)w, (GLsizei)h);           //parameters- (origin bottom-left x of the
viewport, y, width, height of the viewport)

    glMatrixMode(GL_PROJECTION);                        //to change into projection mode

    glLoadIdentity();

    GLfloat aspect = (GLfloat)w / (GLfloat)h;

    if (w >= h) {

        glOrtho(-280.0 * aspect, 280.0 * aspect, -280.0, 280.0, -1400, 400);

    }

    else {

        glOrtho(-280.0, 280.0, -280.0 / aspect, 280.0 / aspect, -1400, 400);

    }

    glMatrixMode(GL_MODELVIEW);                        //to change into modelview mode. you always need to
be inside modelview mode in order to draw

}

void init(){

    glClearColor(0.0,0.4,0.3,1); // parameters- R, G, B, alpha -- to change the background color of the
graphics window

    // glClearColor(.31,.31,.18,1);

    glEnable(GL_DEPTH_TEST);

}

static void key(unsigned char key, int x, int y)

{

    switch (key)

    {

        case 32 :

        case 27 :

        case 'q':

            exit(0);

            break;

        case 'a':

            if(angle<90)angle+=10;

```

```

        break;
case 'd':
    if(angley>0)angley-=10;
    break;
case 'w':
    if(anglex>0)anglex-=10;
    break;
case 's':
    if(anglex<90)anglex+=10;
    break;
case 'j':
    cw-=5;
    break;
case 'l':
    cw+=5;
    break;
case 'i':
    bht+=1;
    break;
case 'k':
    bht-=1;
    break;
case 't':
    st+=1;
    break;
case 'r':
    st-=1;
    break;
case 'Z':
    DLIprod++;
    break;
case 'X':

```

```

DL1prod+=0.1;
    break;
case 'C':
    DL1prod+=0.01;
    break;
case 'M':
    DL2prod++;
    break;
case 'N':
    DL2prod+=0.1;
    break;
case 'B':
    DL2prod+=0.01;
    break;
case 'z':
    DL1prod--;
    break;
case 'x':
    DL1prod-=0.1;
    break;
case 'c':
    DL1prod-=0.01;
    break;
case 'm':
    DL2prod--;
    break;
case 'n':
    DL2prod-=0.1;
    break;
case 'b':
    DL2prod-=0.01;

```

```

        break;

    case '1':
        dip-=1;
        break;
    case '2':
        dip+=1;
        break;
    case '3':
        ha-=1;
        break;
    case '4':
        ha+=1;
        break;
    case 'g':
        recovery-=0.01;
        break;
    case 'h':
        recovery+=0.01;
        break;
}
glutPostRedisplay();
}

void Specialkey(int key, int x, int y)
{
    switch(key)
    {
        case GLUT_KEY_UP:
/*Do whatever you want*/
            a+=10;
            break;

```

```

case GLUT_KEY_DOWN:
/*Do whatever you want*/

    a-=10;

    break;

case GLUT_KEY_LEFT:

    b-=10;

/*Do whatever you want*/

    break;

case GLUT_KEY_RIGHT:

    b+=10;

/*Do whatever you want*/

    break;

}

glutPostRedisplay();

}

int main(int argc, char** argv){

    glutInit(&argc,argv);//To initialise glut library

    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH); //in double buffer mode of
animation, we just redraw the primitive in the back buffer and then swap it with the front buffer, and keep
on doing it. It eliminates the problem of distortion

    glutInitWindowPosition(200,00);

    glutInitWindowSize(1000,800);//window size in width x height format.


    glutCreateWindow("Balancing Diagram for Horizontal Tandem");//Name of the window

    glutDisplayFunc(display);

    glutReshapeFunc(reshape);

    glutKeyboardFunc(key);

    glutSpecialFunc(Specialkey);

    init();

    glutMainLoop();

}

```


APPENDIX-3 Computer program codes of three-dimensional balancing

diagram for vertical tandem operation (mode-1 and mode-2)

```
#include<GL/gl.h>
#include<GL/glu.h>
#include<GL/glut.h>
#include<iostream>
#include<bits/stdc++.h>
#define PI 3.1416
using namespace std;
float anglex=0;
float a=0,b=0;
float angley=0;
float stripping;
float coalRibVol;
int mode=1; //keycut + topcut    mode 2: topcut only
float x,y,z;
//st=seam thickness, bhl= lower bench height, bhu=upper bench height
//cw=cut width, kwt=key cut width top, kwb= key cut width bottom
int LeBucket= 10, LeBoom=70, LaBucket=15, LaBoom=90;
string Le=to_string(LeBucket)+"_"+to_string(LeBoom);
string La=to_string(LaBucket)+"_"+to_string(LaBoom);
//geology
float st=18, dip=0, coal_dens=1.52;
//dragline specifications
// DL24/96 - 3.42/88 , DL15/90 - 2.44,83 , DL10/70 - 1.67, 66
float DL1prod=2.44, reach1=83, DL2prod=3.42, reach2=88;
float dumph=41.5;
//pit design
float cw=70, bht=30, bhl=bht, bhu=bht-bhl; //st- seam thickness, bhl=lower(main) bench
height, bhu- bench height upper
float kwt=35,kwb=5,bermlb=20, bermrh=10.5; // \_/_/ kwt -key cut width top ,kwb- key cut
width bottom
float riba=80,ha=70, ba=60, balg=45, aor=38; //riba- angle of coal extraction, ha- highwall angle, ba-
bench angle, balg- bench angle for loose ground, aor- angle of repose
float recovery=0.9;
float rad= 3.14/180;
```



```

float keycut=0.0, firstdig=0.0, rehandle=0.0, topcut=0.0, firstcut=0.0,areaLead,areaLag;

float prodKC, prodFC, prodTC, prodFD, prodRH, coalExpDL1, coalExpDL2, coalExp, rh_per,
linear_DL1, linear_DL2, DLLeVol, DLLaVol;

float prod_ratio= DL1prod/DL2prod;

string s,s1,v1,v2;

float posx,posy;

float xpos;

string set2decimal(float fl){
    string str;
    std::stringstream stream;
    stream << std::fixed << std::setprecision(2) << fl;
    str = stream.str();
    return str;
}

float capacity(int bucket){
    float cap;
    //swell=39%
    //F=0.933, S=0.719, M=0.75, K=0.73
    //Cap= B/C*F*S*M*K*3600*8760 m3
    if(bucket==10) cap=1.67;    //C=70
    else if(bucket==15) cap=2.44; //72
    else if(bucket==24) cap=3.42; //80
    return cap;
}

int reach(int boom){
    int r;
    if(boom==96) r=88;
    else if(boom==90) r=83;
    else if(boom==70) r=66;
    return r;
}

void output(){
    x=-200;
    y=-80;
    z=0;
    glTranslated(0,a,0);

```

```

glBegin(GL_QUADS); //Colour of the box
    glColor3f(0.1,0.1, 0);
    glVertex3f(x-60,y-170,z-10);
    glVertex3f(-x+50,y-170,z-10);
    glVertex3f(-x+50,y+10,z-10);
//    glColor3f(.35,.35,.67);
    glVertex3f(x-60,y+10,z-10);
glEnd();

```

```

glBegin(GL_LINES); //box lines
glColor4f(0.5,0,0,1);
glVertex3f(x-60,y+10,z);
glVertex3f(-x+50,y+10,z);
glVertex3f(x-60,y-170,z);
glVertex3f(-x+50,y-170,z);
glVertex3f(x-60,y-5,z);
glVertex3f(-x+50,y-5,z);
glVertex3f(x-60,y-170,z);
glVertex3f(x-30,y-170,z);
glVertex3f(x-30,y+10,z);
glVertex3f(x-10,y-170,z);
glVertex3f(x-10,y+10,z);
glVertex3f(-x+50,y-170,z);
glVertex3f(-x+50,y+10,z);
glVertex3f(x+200+10,y-170,z);
glVertex3f(x+200+10,y+10,z);
glEnd();

```

```

glBegin(GL_LINES); //box lines
glColor4f(0.5,0,0,1);
glVertex3f(x-60,y-142,z);
glVertex3f(-x-190,y-143,z);
glEnd();
glColor3f(1,1,1);
s="Input parameters";
glRasterPos3f(x,y,z);

```

```

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

    s=
"Rotation keys for 3-D view";

    glRasterPos3f(x,y+35,z);

    for(int i=0;i<s.length();i++)

        s="Plan view rotation -> a,d";

    glRasterPos3f(x,y+25,z);

    for(int i=0;i<s.length();i++)

        glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

        s="Sectional view rotation -> w,s";

    glRasterPos3f(x,y+15,z);

    for(int i=0;i<s.length();i++)

        glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

    glColor3f(0.9, 0.1, 0.1);

    s="(A)Dragline parameters";

    glRasterPos3f(x,y-12,z);

    for(int i=0;i<s.length();i++)

        glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

    glColor3f(1, 1, 1);

    s="Leading D/L production = " + set2decimal(DL1prod)+ " Mm^3/yr";

    glRasterPos3f(x,y-20,z);

    for(int i=0;i<s.length();i++)

        glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

    s="Lagging D/L production = " + set2decimal(DL2prod)+ " Mm^3/yr";

    glRasterPos3f(x,y-30,z);

    for(int i=0;i<s.length();i++)

        glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

    glColor3f(0.9, 0.1, 0.1);

    s="(B)Operating parameters";

    glRasterPos3f(x,y-40,z);

    for(int i=0;i<s.length();i++)

        glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

    glColor3f(1, 1, 1);

    s="Cut width = " + set2decimal(cw) + " m";

    glRasterPos3f(x,y-50,z);

```

```

for(int i=0;i<s.length();i++)
s="Bench height  = " + set2decimal(bht) + " m";
glRasterPos3f(x,y-60,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="Highwall slope; Repose angle= 70; 38 deg";
glRasterPos3f(x,y-70,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="Bench slope  = " + set2decimal(ba) + " deg";
glRasterPos3f(x,y-80,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="Seam thickness = " + set2decimal(st) + " m";
glRasterPos3f(x,y-130,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="Seam gradient = " + set2decimal(dip) + " deg";
glRasterPos3f(x,y-140,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Inc";
glRasterPos3f(x-50,y,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
glColor3f(1, 1, 1);

s="j";
glRasterPos3f(x-20,y-130,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="l";
glRasterPos3f(x-45,y-130,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

```

```

s="i";

glRasterPos3f(x-45,y-140,z);
for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="k";

glRasterPos3f(x-20,y-140,z);
for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="3";

glRasterPos3f(x-20,y-70,z);
for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="4";

glRasterPos3f(x-45,y-70,z);
for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="v";

glRasterPos3f(x-45,y-50,z);
for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="MNB";

glRasterPos3f(x-50,y-30,z);
for(int i=0;i<s.length();i++)

s="ZXC";

glRasterPos3f(x-50,y-20,z);
for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="e";

glRasterPos3f(x-20,y-50,z);
for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="mnb";

glRasterPos3f(x-28,y-30,z);
for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

```

```

s="zxc";
glRasterPos3f(x-28,y-20,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="h";
glRasterPos3f(x-45,y-100,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="g";
glRasterPos3f(x-20,y-100,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
/*s="5";
glRasterPos3f(x-20,y-80,z);
for(int i=0;i<s.length();i++)
    s="6";
glRasterPos3f(x-45,y-80,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);*/
s="t";
glRasterPos3f(x-45,y-60,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="r";
glRasterPos3f(x-20,y-60,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="Dec";
glRasterPos3f(x-28,y,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="Coal density  = " + set2decimal(coal_dens) + " te/m^3";
glRasterPos3f(x,y-110,z);
for(int i=0;i<s.length();i++)

```

```

glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

glColor3f(0.9, 0.1, 0.1);

s="(C)Geomining parameters";

glRasterPos3f(x,y-120,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

glColor3f(1, 1, 1);

s="Key cut btm w = " + set2decimal(kwb) + " m";

glRasterPos3f(x,y-90,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Coal recovery = " + set2decimal(recovery*100) + " %";

glRasterPos3f(x,y-100,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

if(mode==1){s1="TC + KC";}

else if(mode==2) s1="TC only";

if(mode==1){v1="FD + RH";}

else if(mode==2) v1="KC+FD+RH";

s="Leading D/L -> " + s1 + " (mode : " + to_string(mode) +)";

glRasterPos3f(x,y-150,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Lagging D/L -> " + v1 + " (mode : " + to_string(mode) +)";

glRasterPos3f(x,y-160,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

x=20;

s="Output values";

glRasterPos3f(x,y,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Lower bench height = " + set2decimal(bhl) + " m";

glRasterPos3f(x,y-30,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

```

```

s="Upper bench height  = " + set2decimal(bhu) + " m";
glRasterPos3f(x,y-20,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="Stripping ratio (SR) = " + set2decimal(stripping);
glRasterPos3f(x,y-12,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="Top cut area(T/C)  = " + set2decimal(topcut) + " m^2";
glRasterPos3f(x,y-40,z);
for(int i=0;i<s.length();i++)

s="Key cut area(K/C)   = " + set2decimal(keycut) + " m^2";
glRasterPos3f(x,y-50,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="First dig area(F/D) = " + set2decimal(firstdig) + " m^2";
glRasterPos3f(x,y-60,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="Rehandled area(R/H) = " + set2decimal(rehandle) + " m^2";
glRasterPos3f(x,y-70,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="Lead D/L cut area   = " + set2decimal(areaLead) + " m^2";
glRasterPos3f(x,y-80,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
s="Lag D/L cut area    = " + set2decimal(areaLag) + " m^2";
glRasterPos3f(x,y-90,z);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Lead D/L cut vol     = " + set2decimal(DLLeVol) + " Mm^3/yr";
glRasterPos3f(x,y-100,z);

```



```

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Lag D/L cut vol    = " + set2decimal(DLLaVol) + " Mm^3/yr";

glRasterPos3f(x,y-110,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Coal exposure     = " + set2decimal(coalExpDL1) + " Mte/yr";

glRasterPos3f(x,y-120,z);

for(int i=0;i<s.length();i++)

s="Rehandle percentage  = " + set2decimal(rh_per) + " %";

glRasterPos3f(x,y-130,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Lead D/L linear adv  = " + set2decimal(linear_DL1) + " m/yr";

glRasterPos3f(x,y-140,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Lag D/L linear adv   = " + set2decimal(linear_DL2) + " m/yr";

glRasterPos3f(x,y-150,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

s="Coal rib volume     = " + set2decimal(coalRibVol/1000000) + " Mm^3/yr";

glRasterPos3f(x,y-160,z);

for(int i=0;i<s.length();i++)

    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

}

void display(){

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glLoadIdentity();

    bhl=bht,bhu=bht-bhl;

//  DL1prod=capacity(LeBucket);

//  DL2prod=capacity(LaBucket);

    reach1=reach(LeBoom);

    reach2=reach(LaBoom);

    prod_ratio= DL1prod/DL2prod;

```

```

float x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16,x17,x18,dx;

float ribx, riby;

float xd[15],yd[15];

x1=0; //A
x2=cw; //B
x3=x2+cw; //B'
y1=0;
y2=-x2*tan(dip*rad);
y3=-x3*tan(dip*rad);
dx=st*cos(dip*rad)/sin((dip+riba)*rad)*cos(riba*rad);
dy=st*cos(dip*rad)/sin((dip+riba)*rad)*sin(riba*rad);
x4=x1+dx; //C
x5=x2+dx; //D
x6=x3; //D'
y4=y1+dy;
y5=y2+dy;
y6=y3+st;
dx=bhl*cos(dip*rad)/sin((dip+ha)*rad)*cos(ha*rad);

x7=x4+dx; //E
x8=x5+dx; //F
x9=x6; //F'
y7=y4+dy;
y8=y5+dy;
y9=y6+bhl;
x10=x7+bermlb;//*cos(dip*rad); //E' - upper bench bottom left
y10=y7-bermlb*tan(dip*rad);
x11=x10+cw; //J' - upper bench bottom right
y11=y10-cw*tan(dip*rad);
dx=bhu*cos(dip*rad)/sin((dip+ha)*rad)*cos(ha*rad);
dy=bhu*cos(dip*rad)/sin((dip+ha)*rad)*sin(ha*rad);
x13=x10+dx; //E'' - upper bench top left
y13=y10+dy;
x14=x11+dx; //J'' - upper bench top right
y14=y11+dy;

```

```

x12=x9; //J'''
y12=y9+bhu;

dx=bh1*cos(dip*rad)/sin((dip+(180-ba))*rad)*cos((180-ba)*rad);
x15=x5-kwb; //H
//x16=x8-kwt; //J
x16=x15+dx;
y15=y5+kwb*tan(dip*rad);
//y16=y8+kwt*tan(dip*rad);
y16=y15+dy;
posx=(x16-x7)/2;
posy=posx*tan(dip*rad);
x17=x15-posx; //G
x18=x16-posx; //I
y17=y15+posy;
y18=y16+posy;
//dump side
xd[0]=x1-cw; //T
yd[0]=-xd[0]*tan(dip*rad);
xd[1]=x1-2*cw; //T'
yd[1]=-xd[1]*tan(dip*rad);
xd[2]=xd[1]; //T''
yd[2]=y7;
xd[3]=(x8+x16)/2 - reach1; //L
yd[3]=y7;
xd[4]= ((tan(aor*rad)*xd[3] - tan((180-dip)*rad)*x4)-(yd[3]-y4))/(tan(aor*rad)-tan((180-dip)*rad));
// K
yd[4]= xd[4]*tan(aor*rad)+yd[3]-tan(aor*rad)*xd[3];
xd[5]=x4-bermrh; //O
yd[5]=y4+bermrh*tan(dip*rad);
xd[6]=(x16+x18)/2 - reach1; //N
yd[6]=y7;
xd[7]= ((tan(aor*rad)*xd[6] - tan((180-dip)*rad)*x4)-(yd[6]-y4))/(tan(aor*rad)-tan((180-dip)*rad));
//M
yd[7]= xd[7]*tan(aor*rad)+yd[6]-tan(aor*rad)*xd[6];
xd[8]= (y7-(yd[5]-tan((180-balg)*rad)*xd[5]))/tan((180-balg)*rad); //P
yd[8]= y7;

```

```

xd[9]= xd[8]-bermrh-5; //S
yd[9]=y7;
yd[10]=y7+dumph; //Q
xd[10]=xd[9]- dumph/tan(rad*aor);
xd[11]= xd[10]-(xd[9]-xd[10]); //R
yd[11]= y7;
ribx=2*st/tan(riba*rad);
riby= ribx*tan(dip*rad);
//optimization
posx=0;
firstdig=0.5*((x15-x4)+(x16-x7))*bhl; /*cos(dip*rad);
topcut= 0.5*((x11-x10)+(x14-x13))*bhu;
/*x= ((tan(aor*rad)*xd[6] - tan((180-0)*rad)*x4)-(yd[6]-y4))/(tan(aor*rad)-tan((180-0)*rad));
y= x*tan(aor*rad)+yd[6]-tan(aor*rad)*xd[6];
a=0.5*((x7-xd[6])+(x4-x))*(y7-y4);
b=0.5*(xd[5]-x)*((xd[5]-x)/2)*tan(aor*rad);
c=0.5*((x4-xd[5])+(x7-xd[8]))*(y7-y4);
rehandle=a-b;*/
rehandle= 0.5*(bermrh+ (x7-xd[8]))*bhl;
rehandle=1.1*rehandle;
if(mode==1){
    areaLead=keycut+topcut;
    areaLag=firstdig+rehandle;
}
if(mode==2){
    areaLead=topcut;
    areaLag=keycut+firstdig+rehandle;
}
if(areaLag >= (areaLead/prod_ratio)){
    while(areaLag >= (areaLead/prod_ratio)){
        bhl-=0.0001;
        bhu=bht-bhl;
        dx=bhl*cos(dip*rad)/sin((dip+ha)*rad)*cos(ha*rad);
        dy=bhl*cos(dip*rad)/sin((dip+ha)*rad)*sin(ha*rad);
        x7=x4+dx; //E

```

```

x8=x5+dx; //F
x9=x6; //F'
y7=y4+dy;
y8=y5+dy;
y9=y6+bhl;
x10=x7+bermlb;/*cos(dip*rad); //E' - upper bench bottom left
y10=y7-bermlb*tan(dip*rad);
x11=x10+cw; //J' - upper bench bottom right
y11=y10-cw*tan(dip*rad);

dx=bhu*cos(dip*rad)/sin((dip+ha)*rad)*cos(ha*rad);
dy=bhu*cos(dip*rad)/sin((dip+ha)*rad)*sin(ha*rad);
x13=x10+dx; //E'' - upper bench top left
y13=y10+dy;
x14=x11+dx; //J'' - upper bench top right
y14=y11+dy;
x12=x9; //J'''
y12=y9+bhu;
dx=bhl*cos(dip*rad)/sin((dip+(180-ba))*rad)*cos((180-ba)*rad);
x15=x5-kwb; //H
//x16=x8-kwt; //J
x16=x15+dx;
y15=y5+kwb*tan(dip*rad);
//y16=y8+kwt*tan(dip*rad);
y16=y15+dy;
posx=(x16-x7)/2;
posy=posx*tan(dip*rad);
x17=x15-posx; //G
x18=x16-posx; //I
y17=y15+posy;
y18=y16+posy;
//dump side
xd[0]=x1-cw; //T
yd[0]=-xd[0]*tan(dip*rad);
xd[1]=x1-2*cw; //T'

```

```

yd[1]=-xd[1]*tan(dip*rad);

xd[2]=xd[1]; //T"

yd[2]=y7;

xd[3]=(x8+x16)/2 - reach1; //L

yd[3]=y7;

xd[4]= ((tan(aor*rad)*xd[3] --dip)*rad)); // K

yd[4]= xd[4]*tan(aor*rad)+yd[3]-tan(aor*rad)*xd[3];

xd[5]=x4-bermrh; //O

yd[5]=y4+bermrh*tan(dip*rad);

xd[6]= (x16+x18)/2 - reach1; //N

yd[6]=y7;

xd[7]= ((tan(aor*rad)*xd[6] - tan((180-dip)*rad)*x4)-(yd[6]-y4))/(tan(aor*rad)-tan((180-
dip)*rad)); //M

yd[7]= xd[7]*tan(aor*rad)+yd[6]-tan(aor*rad)*xd[6];

xd[8]= (y7-(yd[5]-tan((180-balg)*rad)*xd[5]))/tan((180-balg)*rad); //P

yd[8]= y7;

xd[9]= xd[8]-bermrh-5; //S

yd[9]=y7;

yd[10]=y7+dumph; //Q

xd[10]=xd[9]- dumph/tan(rad*aor);

xd[11]= xd[10]-(xd[9]-xd[10]); //R

yd[11]= y7;

keycut=0.5*((x5-x15)+(x8-x16))*bhl;/*cos(dip*rad);

firstdig=0.5*((x15-x4)+(x16-x7))*bhl;/*cos(dip*rad);

topcut= 0.5*((x11-x10)+(x14-x13))*bhu;

/*x= ((tan(aor*rad)*xd[6] - tan((180-0)*rad)*x4)-(yd[6]-y4))/(tan(aor*rad)-tan((180-0)*rad));

y= x*tan(aor*rad)+yd[6]-tan(aor*rad)*xd[6];

a=0.5*((x7-xd[6])+(x4-x))*(y7-y4);

b=0.5*(xd[5]-x)*((xd[5]-x)/2)*tan(aor*rad);

c=0.5*((x4-xd[5])+(x7-xd[8]))*(y7-y4);

rehandle=a-b;*/

rehandle= 0.5*(bermrh+ (x7-xd[8]))*bhl;

rehandle=1.1*rehandle;

if(mode==1){

    areaLead=keycut+topcut;

```

```

areaLag=firstdig+rehandle;
    }
    if(mode==2){
        areaLead=topcut;
        areaLag=keycut+firstdig+rehandle;
    }
}
}
/*if(areaLag < (areaLead/prod_ratio)){
    while(areaLag < (areaLead/prod_ratio)){
        bhl+=0.0001;
        bhu=bht-bhl;
        dx=bhl*cos(dip*rad)/sin((dip+ha)*rad)*cos(ha*rad);
        x7=x4+dx;                                //E
        x8=x5+dx;                                //F
        x9=x6;                                    //F'
        y7=y4+dy;
        y8=y5+dy;
        y9=y6+bhl;
        x10=x7+bermlb;/*cos(dip*rad);                //E' - upper bench bottom left
        y10=y7-bermlb*tan(dip*rad);
        x11=x10+cw;                                //J' - upper bench bottom right
        y11=y10-cw*tan(dip*rad);
        dx=bhu*cos(dip*rad)/sin((dip+ha)*rad)*cos(ha*rad);
        dy=bhu*cos(dip*rad)/sin((dip+ha)*rad)*sin(ha*rad);
        x13=x10+dx;                                //E'' - upper bench top left
        y13=y10+dy;
        x14=x11+dx;                                //J'' - upper bench top right
        y14=y11+dy;
        x12=x9;                                    //J'''
        y12=y9+bhu;
        dx=bhl*cos(dip*rad)/sin((dip+(180-ba))*rad)*cos((180-ba)*rad);
        dy=bhl*cos(dip*rad)/sin((dip+(180-ba))*rad)*sin((180-ba)*rad);
        x15=x5-kwb;                                //H
        //x16=x8-kwt;                                //J

```

```

x16=x15+dx;
y15=y5+kwb*tan(dip*rad);
//y16=y8+kwt*tan(dip*rad);
y16=y15+dy;

posx=(x16-x7)/2;
posy=posx*tan(dip*rad);
x17=x15-posx; //G
x18=x16-posx; //I
y17=y15+posy;
y18=y16+posy;
//dump side
xd[0]=x1-cw; //T
yd[0]=-xd[0]*tan(dip*rad);
xd[1]=x1-2*cw; //T'
yd[1]=-xd[1]*tan(dip*rad);
xd[2]=xd[1]; //T''
yd[2]=y7;
xd[3]=(x8+x16)/2 - reach1; //L
yd[3]=y7;
yd[4]= xd[4]*tan(aor*rad)+yd[3]-tan(aor*rad)*xd[3];
xd[5]=x4-bermrh; //O
yd[5]=y4+bermrh*tan(dip*rad);
xd[6]= (x16+x18)/2 - reach1; //N
yd[6]=y7;
xd[7]= ((tan(aor*rad)*xd[6] - tan((180-dip)*rad)*x4)-(yd[6]-y4))/(tan(aor*rad)-tan((180-
dip)*rad)); //M
yd[7]= xd[7]*tan(aor*rad)+yd[6]-tan(aor*rad)*xd[6];
xd[8]= (y7-(yd[5]-tan((180-balg)*rad)*xd[5]))/tan((180-balg)*rad); //P
yd[8]= y7;
xd[9]= xd[8]-bermrh-5; //S
yd[9]=y7;
yd[10]=y7+dumph; //Q
xd[10]=xd[9]- dumph/tan(rad*aor);
xd[11]= xd[10]-(xd[9]-xd[10]); //R
yd[11]= y7;

```



```

keycut=0.5*((x5-x15)+(x8-x16))*bhl;/*cos(dip*rad);
firstdig=0.5*((x15-x4)+(x16-x7))*bhl;/*cos(dip*rad);
topcut= 0.5*((x11-x10)+(x14-x13))*bhu;
y= x*tan(aor*rad)+yd[6]-tan(aor*rad)*xd[6];
a=0.5*((x7-xd[6])+(x4-x))*(y7-y4);
b=0.5*(xd[5]-x)*((xd[5]-x)/2)*tan(aor*rad);
c=0.5*((x4-xd[5])+(x7-xd[8]))*(y7-y4);
rehandle=a-b;
rehandle= 0.5*(bermrh+ (x7-xd[8]))*bhl;
rehandle=1.1*rehandle;
if(mode==1){
    areaLead=keycut+topcut;
    areaLag=firstdig+rehandle;
}
if(mode==2){
    areaLead=topcut;
    areaLag=keycut+firstdig+rehandle;
}
}
}*/          //optimization ends

prodKC= keycut/areaLead*DL1prod;
prodTC= topcut/areaLead*DL1prod;
prodFD= firstdig/areaLag*DL2prod;
prodRH= rehandle/areaLag*DL2prod;
linear_DL1= DL1prod*1000000/areaLead;
linear_DL2= DL2prod*1000000/areaLag;
coalExpDL1= linear_DL1*cw*st*coal_dens*recovery/1000000;
coalExpDL2= linear_DL2*cw*st*coal_dens*recovery/1000000;
//coalExp= coalExpDL1+coalExpDL2;
rh_per= rehandle*100/(keycut+topcut+firstdig);
DLLeVol= linear_DL1*areaLead/1000000;
DLLaVol= linear_DL2*areaLag/1000000;
coalRibVol=((ribx*st)/2)*linear_DL1;
stripping=((keycut+topcut+firstdig)*linear_DL1)/(coalExpDL1*1000000);

```

```

    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA); glEnable( GL_BLEND );
    glClearColor(0.0,0.4,0.3,1);

    //display the values

    /*string str;

    std::stringstream stream;

    stream << std::fixed << std::setprecision(2) << keycut;

    str = stream.str();*/

    output();

    glTranslated(0,20,0);

    glRotatef(anglex,1.0,0.0,0.0);

    glRotatef(angley,0.0,1.0,0.0);

    //draw

    glColor3f(1,1,1);

    s= "R/H";

    glRasterPos3f((xd[8]+x7)/2,(y4+y7)/2,70);

    for(int i=0;i<s.length();i++)

        glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

    s= "Highwall";

    glRasterPos3f((x16+x8)/2-2+20,(y15+y16)/2,50);

    for(int i=0;i<s.length();i++)

        glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

    s= "Mine advancement";

    for(int i=0;i<s.length();i++)

        glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

    s= "----->";

    glRasterPos3f((x16+x8)/2-2+40,(y15+y16)/2+30,50);

    for(int i=0;i<s.length();i++)

        glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

    s= "Spoil side/Decoaled area";

    glRasterPos3f((x16+x8)/2-2+20-300,(y15+y16)/2-10,50);

    for(int i=0;i<s.length();i++)

        glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);

    s= "Overburden dumps";

    glRasterPos3f((x16+x8)/2-2+20-300,(y15+y16)/2+68,50);

    for(int i=0;i<s.length();i++)

```

```

        glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
        s= "3-D Balancing diagram for vertical tandem";
glRasterPos3f(x-150,y+245,50);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
glBegin(GL_QUADS);                                //rehandle
glColor4f(0.33,0.06,0.2,0.8);
glVertex3f(x4,y4,70);
glVertex3f(x7,y7,70);
glVertex3f(xd[8],yd[8],70);
glVertex3f(xd[5],yd[5],70);

glVertex3f(x4,y4,0);
glVertex3f(x7,y7,0);
glVertex3f(xd[8],yd[8],0);
glVertex3f(xd[5],yd[5],0);
glVertex3f(x4,y4,70);
glVertex3f(x7,y7,70);
glVertex3f(x7,y7,00);
glVertex3f(x4,y4,00);
glVertex3f(x7,y7,70);
glVertex3f(xd[8],yd[8],70);
glVertex3f(xd[8],yd[8],00);
glVertex3f(x7,y7,00);
glVertex3f(xd[8],yd[8],70);
glVertex3f(xd[5],yd[5],70);
glVertex3f(xd[5],yd[5],00);
glVertex3f(xd[8],yd[8],00);
glVertex3f(xd[5],yd[5],70);
glVertex3f(x4,y4,70);
glVertex3f(x4,y4,00);
glVertex3f(xd[5],yd[5],00);
glEnd();

/* glBegin(GL_QUADS);                                //rehandle 1
glColor4f(0.33,0.06,0.2,0.8);

```

```

glVertex3f(xd[4],yd[4],70);
glVertex3f(x4,y4,70);
glVertex3f(x7,y7,70);
glVertex3f(xd[3],yd[3],70);
glVertex3f(xd[4],yd[4],70);
glVertex3f(x4,y4,70);
glVertex3f(x4,y4,00);
glVertex3f(xd[4],yd[4],00);
glVertex3f(x4,y4,70);
glVertex3f(x7,y7,70);
glVertex3f(x7,y7,00);
glVertex3f(x4,y4,00);
glVertex3f(x7,y7,70);
glVertex3f(xd[3],yd[3],70);
glVertex3f(xd[3],yd[3],00);
glVertex3f(x7,y7,00);
glVertex3f(xd[3],yd[3],70);
glVertex3f(xd[4],yd[4],70);
glVertex3f(xd[4],yd[4],00);
glVertex3f(xd[3],yd[3],00);
glVertex3f(xd[4],yd[4],00);
glVertex3f(x4,y4,00);
glVertex3f(x7,y7,00);
glVertex3f(xd[3],yd[3],00);
glEnd();
glBegin(GL_QUADS);                                //rehandle 2
glColor4f(1.0,0,1,0.8);
glVertex3f(xd[7],yd[7],70);
glVertex3f(xd[4],yd[4],70);
glVertex3f(xd[3],yd[3],70);
glVertex3f(xd[6],yd[6],70);
glVertex3f(xd[7],yd[7],70);
glVertex3f(xd[4],yd[4],70);
glVertex3f(xd[4],yd[4],00);
glVertex3f(xd[7],yd[7],00);
glVertex3f(xd[4],yd[4],70);

```

```

glVertex3f(xd[3],yd[3],70);
glVertex3f(xd[3],yd[3],00);
glVertex3f(xd[4],yd[4],00);
glVertex3f(xd[3],yd[3],70);
glVertex3f(xd[6],yd[6],70);
glVertex3f(xd[6],yd[6],00);
glVertex3f(xd[3],yd[3],00);
glVertex3f(xd[6],yd[6],70);
glVertex3f(xd[7],yd[7],70);
glVertex3f(xd[7],yd[7],00);
glVertex3f(xd[6],yd[6],00);
glVertex3f(xd[7],yd[7],70);
glVertex3f(xd[4],yd[4],70);
glVertex3f(xd[3],yd[3],70);
glVertex3f(xd[6],yd[6],70);
glEnd(); */

glBegin(GL_TRIANGLES);                                //coal rib
glColor3f(0,0,0);
glVertex3f(xd[0],yd[0],150);
glVertex3f(xd[0]+ribx, yd[0]-riby,150);
glVertex3f(xd[0]+ribx/2, yd[0]-riby/2+st,150);
glEnd();

glBegin(GL_QUADS);
glVertex3f(xd[0],yd[0],150);
glVertex3f(xd[0]+ribx, yd[0]-riby,150);
glVertex3f(xd[0]+ribx, yd[0]-riby,-100);
glVertex3f(xd[0],yd[0],-100);

glVertex3f(xd[0]+ribx, yd[0]-riby,150);
glVertex3f(xd[0]+ribx/2, yd[0]-riby/2+st,150);
glVertex3f(xd[0]+ribx/2, yd[0]-riby/2+st,-100);
glVertex3f(xd[0],yd[0],150);
glVertex3f(xd[0]+ribx/2, yd[0]-riby/2+st,150);
glVertex3f(xd[0]+ribx/2, yd[0]-riby/2+st,-100);
glVertex3f(xd[0],yd[0],-100);
glEnd();

```

```

glBegin(GL_TRIANGLES);                                //coal rib 2
glColor3f(0,0,0);
glVertex3f(xd[1],yd[1],150);
glVertex3f(xd[1]+ribx, yd[1]-riby,150);
glVertex3f(xd[1]+ribx/2, yd[1]-riby/2+st,150);
glEnd();
glBegin(GL_QUADS);
glVertex3f(xd[1],yd[1],150);
glVertex3f(xd[1]+ribx, yd[1]-riby,150);
glVertex3f(xd[1]+ribx, yd[1]-riby,-100);
glVertex3f(xd[1],yd[1],-100);
glVertex3f(xd[1]+ribx, yd[1]-riby,150);
glVertex3f(xd[1]+ribx/2, yd[1]-riby/2+st,-100);
glVertex3f(xd[1]+ribx, yd[1]-riby,-100);
glVertex3f(xd[1],yd[1],150);
glVertex3f(xd[1]+ribx/2, yd[1]-riby/2+st,150);
glVertex3f(xd[1]+ribx/2, yd[1]-riby/2+st,-100);
glVertex3f(xd[1],yd[1],-100);
glEnd();
x=-2*cw- 20;
if(x> xd[11]-10) x=xd[11]-10;
glBegin(GL_POLYGON);                                //dump
glColor4f(1.0,1.0,0.0,0.8);
glVertex3f(x1,y1,150);
glVertex3f(x-70,-x*tan(rad*dip),150);
glVertex3f(x-70,yd[2],150);
glVertex3f(xd[8],yd[8],150);
glVertex3f(xd[5],yd[5],150);
glVertex3f(x4,y4,150);
glEnd();
glBegin(GL_QUADS);
glVertex3f(x1,y1,150);
glVertex3f(x-70,-x*tan(rad*dip),150);
glVertex3f(x-70,-x*tan(rad*dip),z-100);
glVertex3f(x1,y1,z-100);
glVertex3f(x-70,-x*tan(rad*dip),150);

```

```

glVertex3f(x-70,yd[2],150);
glVertex3f(x-70,yd[2],z-100);
glVertex3f(x-70,-x*tan(rad*dip),z-100);
glVertex3f(x-70,yd[2],150);
glVertex3f(xd[8],yd[8],150);
glVertex3f(xd[8],yd[8],z-100);
glVertex3f(x-70,yd[2],z-100);

glVertex3f(xd[8],yd[8],150);
glVertex3f(xd[5],yd[5],150);
glVertex3f(xd[5],yd[5],z-100);
glVertex3f(xd[8],yd[8],z-100);
glVertex3f(xd[5],yd[5],150);
glVertex3f(x4,y4,150);
glVertex3f(x4,y4,z-100);
glVertex3f(xd[5],yd[5],z-100);
glEnd();

glBegin(GL_TRIANGLES);                                //dump triangle
glColor4f(0.55,0.46,0.05,0.95);
glVertex3f(xd[9],yd[9],150);
glVertex3f(xd[10],yd[10],150);
glVertex3f(xd[11],yd[11],150);
glEnd();

glBegin(GL_TRIANGLES);                                //dump triangle 2 by vaibhav
glColor4f(0.55,0.46,0.05,0.95);
glVertex3f(xd[9]-70,yd[9],150);
glVertex3f(xd[10]-70,yd[10],150);
glEnd();

glBegin(GL_QUADS);
glVertex3f(xd[9],yd[9],150);
glVertex3f(xd[10],yd[10],150);
glVertex3f(xd[10],yd[10],0);
glVertex3f(xd[9],yd[9],0);
glVertex3f(xd[10],yd[10],150);
glVertex3f(xd[11],yd[11],150);
glVertex3f(xd[11],yd[11],0);

```

```

glVertex3f(xd[10],yd[10],0);
glVertex3f(xd[9],yd[9],150);
glVertex3f(xd[11],yd[11],150);
glVertex3f(xd[11],yd[11],0);
glVertex3f(xd[9],yd[9],0);
glEnd();

glBegin(GL_QUADS); // dump triangle 2 3d by vaibhav
glVertex3f(xd[9]-70,yd[9],150);
glVertex3f(xd[10]-70,yd[10],150);
glVertex3f(xd[10]-70,yd[10],0);
glVertex3f(xd[9]-70,yd[9],0);

glVertex3f(xd[10]-70,yd[10],150);
glVertex3f(xd[11]-70,yd[11],150);
glVertex3f(xd[11]-70,yd[11],0);
glVertex3f(xd[10]-70,yd[10],0);
glVertex3f(xd[9]-70,yd[9],150);
glVertex3f(xd[11]-70,yd[11],150);
glVertex3f(xd[11]-70,yd[11],0);
glVertex3f(xd[9]-70,yd[9],0);
glEnd();

glBegin(GL_TRIANGLES);
glColor4f(0.55,0.46,0.0,0.95);
glVertex3f(xd[9],yd[9],00);
glVertex3f(xd[10],yd[10],00);
glVertex3f(xd[11],yd[11],00);
glEnd();

glBegin(GL_TRIANGLES); //dump triangle 2
glColor4f(0.55,0.46,0.05,0.95);
glVertex3f(xd[9],yd[9],150);
glVertex3f(xd[10],yd[10],150);
glEnd();

glBegin(GL_QUADS);
glVertex3f(xd[9],yd[9],150);

```



```

glVertex3f(xd[10],yd[10],150);
glVertex3f(xd[10],yd[10],0);
glVertex3f(xd[9],yd[9],0);
glVertex3f(xd[10],yd[10],150);
glVertex3f(xd[11],yd[11],150);
glVertex3f(xd[11],yd[11],0);
glVertex3f(xd[9],yd[9],150);
glVertex3f(xd[11],yd[11],150);
glVertex3f(xd[11],yd[11],0);
glVertex3f(xd[9],yd[9],0);
glEnd();
glBegin(GL_TRIANGLES);
glColor4f(0.55,0.46,0.0,0.95);
glVertex3f(xd[9],yd[9],00);
glVertex3f(xd[10],yd[10],00);
glVertex3f(xd[11],yd[11],00);
glEnd();
glColor3f(1, 1, 1);
//bench side

```

```

glBegin(GL_QUADS);                                //coal seam part1
glColor4f(0.0,0.0,0.0,1);
glVertex3f(x1,y1,150);
glVertex3f(x2,y2,150);
glVertex3f(x5,y5,150);
glVertex3f(x4,y4,150);
glVertex3f(x1,y1,-100);
glVertex3f(x2,y2,-100);
glVertex3f(x5,y5,-100);
glVertex3f(x4,y4,-100);
glVertex3f(x1,y1,150);
glVertex3f(x4,y4,150);
glVertex3f(x4,y4,-100);
glVertex3f(x1,y1,-100);
glVertex3f(x2,y2,150);

```

```

glVertex3f(x5,y5,150);
glVertex3f(x5,y5,-100);
glVertex3f(x2,y2,-100);
glVertex3f(x1,y1,150);
glVertex3f(x2,y2,150);
glVertex3f(x2,y2,-100);
glVertex3f(x1,y1,-100);
glVertex3f(x4,y4,150);
glVertex3f(x5,y5,150);
glVertex3f(x5,y5,-100);
glVertex3f(x4,y4,-100);
glEnd();

```

```

glBegin(GL_QUADS);                                //coal seam part 2
glColor4f(0.0,0,0.0,1);
//front
glVertex3f(x2,y2,150);
glVertex3f(x3,y3,150);
glVertex3f(x6,y6,150);
glVertex3f(x5,y5,150);
//back
glVertex3f(x2,y2,-100);
glVertex3f(x3,y3,-100);
glVertex3f(x6,y6,-100);
glVertex3f(x5,y5,-100);
//left
glVertex3f(x2,y2,150);
glVertex3f(x5,y5,150);
glVertex3f(x5,y5,-100);
glVertex3f(x2,y2,-100);
//right
glVertex3f(x3,y3,150);
glVertex3f(x6,y6,150);
glVertex3f(x6,y6,-100);
glVertex3f(x3,y3,-100);
//bottom

```

```

glVertex3f(x2,y2,150);
glVertex3f(x3,y3,150);
glVertex3f(x3,y3,-100);
glVertex3f(x2,y2,-100);
//top
glVertex3f(x5,y5,150);
glVertex3f(x6,y6,150);
glVertex3f(x6,y6,-100);
glVertex3f(x5,y5,-100);
glEnd();
glColor3f(1,1,1);
s= "F/D";
glRasterPos3f((x16+x7)/2,(y4+y7)/2,70);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
glBegin(GL_QUADS);                                //first dig
glColor4f(1.0,0.0,0.0,0.7);
glVertex3f(x4,y4,70);
glVertex3f(x17,y17,70);
glVertex3f(x18,y18,70);
glVertex3f(x7,y7,70);
glVertex3f(x17,y17,-100);
glVertex3f(x18,y18,-100);
glVertex3f(x7,y7,-100);

glVertex3f(x4,y4,70);
glVertex3f(x7,y7,70);
glVertex3f(x7,y7,-100);
glVertex3f(x4,y4,-100);
/* glVertex3f(x17,y17,70);
glVertex3f(x18,y18,70);
glVertex3f(x18,y18,-100);
glVertex3f(x4,y4,70);
glVertex3f(x17,y17,70);
glVertex3f(x17,y17,-100);
glVertex3f(x4,y4,-100);

```

```

glVertex3f(x7,y7,70);
glVertex3f(x18,y18,70);
glVertex3f(x18,y18,-100);
glVertex3f(x7,y7,-100);
glEnd();

glBegin(GL_QUADS);           //first cut
glColor4f(1.0,0.0,0.0,0.7);
glVertex3f(x17,y17,70);
glVertex3f(x15,y15,70);
glVertex3f(x16,y16,70);
glVertex3f(x18,y18,70);
glVertex3f(x17,y17,-100);
glVertex3f(x15,y15,-100);
glVertex3f(x16,y16,-100);
glVertex3f(x18,y18,-100);
/* glVertex3f(x17,y17,70);
glVertex3f(x18,y18,70);
glVertex3f(x17,y17,-100); */
glVertex3f(x15,y15,70);
glVertex3f(x16,y16,70);
glVertex3f(x16,y16,-100);
glVertex3f(x15,y15,-100);
glVertex3f(x17,y17,70);
glVertex3f(x15,y15,70);
glVertex3f(x15,y15,-100);
glVertex3f(x17,y17,-100);

glVertex3f(x18,y18,70);
glVertex3f(x16,y16,70);
glVertex3f(x16,y16,-100);
glVertex3f(x18,y18,-100);
glEnd();
glColor3f(1,1,1);
s= "K/C";
glRasterPos3f((x8+x16)/2,(y16+y5)/2,70);

```

```

for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
glBegin(GL_QUADS);                                //key cut
glColor4f(0.0,0.6,1.0,0.7);
glVertex3f(x15,y15,70);
glVertex3f(x5,y5,70);
glVertex3f(x8,y8,70);
glVertex3f(x16,y16,70);
glVertex3f(x15,y15,-100);
glVertex3f(x5,y5,-100);
glVertex3f(x8,y8,-100);
glVertex3f(x16,y16,-100);

glVertex3f(x15,y15,70);
glVertex3f(x16,y16,70);
glVertex3f(x16,y16,-100);
glVertex3f(x15,y15,-100);
glVertex3f(x5,y5,70);
glVertex3f(x8,y8,70);
glVertex3f(x5,y5,-100);
glVertex3f(x15,y15,70);
glVertex3f(x5,y5,70);
glVertex3f(x5,y5,-100);
glVertex3f(x15,y15,-100);
glVertex3f(x16,y16,70);
glVertex3f(x8,y8,70);
glVertex3f(x8,y8,-100);
glVertex3f(x16,y16,-100);
glEnd();

glBegin(GL_QUADS);                                //lower bench part 3
glColor4f(0.8,0.8,0.8,0.9);
glVertex3f(x5,y5,150);
glVertex3f(x6,y6,150);
glVertex3f(x9,y9,150);
glVertex3f(x8,y8,150);

```

```

glVertex3f(x5,y5,-100);
glVertex3f(x6,y6,-100);
glVertex3f(x9,y9,-100);
glVertex3f(x8,y8,-100);
glVertex3f(x5,y5,150);
glVertex3f(x8,y8,150);
glVertex3f(x8,y8,-100);
glVertex3f(x5,y5,-100);
glVertex3f(x6,y6,150);
glVertex3f(x9,y9,150);
glVertex3f(x9,y9,-100);
glVertex3f(x6,y6,-100);
glVertex3f(x5,y5,150);
glVertex3f(x6,y6,150);
glVertex3f(x6,y6,-100);
glVertex3f(x5,y5,-100);
glVertex3f(x8,y8,150);
glVertex3f(x9,y9,150);
glVertex3f(x9,y9,-100);
glVertex3f(x8,y8,-100);
glEnd();
glColor3f(1,1,1);
s= "T/C";
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_8_BY_13,s[i]);
glBegin(GL_QUADS);           //top cut
glColor4f(0.0,0.8,0.0,0.7);
glVertex3f(x10,y10,50);
glVertex3f(x11,y11,50);
glVertex3f(x14,y14,50);
glVertex3f(x13,y13,50);
glVertex3f(x10,y10,-100);
glVertex3f(x11,y11,-100);
glVertex3f(x14,y14,-100);
glVertex3f(x13,y13,-100);

```

```

glVertex3f(x10,y10,50);
glVertex3f(x13,y13,50);
glVertex3f(x13,y13,-100);
glVertex3f(x10,y10,-100);
glVertex3f(x11,y11,50);
glVertex3f(x14,y14,50);
glVertex3f(x14,y14,-100);
glVertex3f(x11,y11,-100);
glVertex3f(x10,y10,50);
glVertex3f(x11,y11,50);
glVertex3f(x11,y11,-100);
glVertex3f(x10,y10,-100);
glVertex3f(x13,y13,50);
glVertex3f(x14,y14,50);
glVertex3f(x14,y14,-100);
glVertex3f(x13,y13,-100);

```

```
glEnd();
```

```

glBegin(GL_QUADS);           //upper bench part 2
glColor4f(0.8,0.8,0.8,0.9);
glVertex3f(x11,y11,150);
glVertex3f(x9,y9,150);
glVertex3f(x12,y12,150);
glVertex3f(x14,y14,150);
glVertex3f(x11,y11,-100);
glVertex3f(x9,y9,-100);
glVertex3f(x12,y12,-100);
glVertex3f(x14,y14,-100);
glVertex3f(x11,y11,150);
glVertex3f(x14,y14,150);
glVertex3f(x11,y11,-100);
glVertex3f(x9,y9,150);
glVertex3f(x12,y12,150);
glVertex3f(x12,y12,-100);

```

```

glVertex3f(x9,y9,-100);
glVertex3f(x11,y11,150);
glVertex3f(x9,y9,150);
glVertex3f(x9,y9,-100);
glVertex3f(x11,y11,-100);
glVertex3f(x14,y14,150);
glVertex3f(x12,y12,150);
glVertex3f(x12,y12,-100);
glVertex3f(x14,y14,-100);
glEnd();

// glBegin(GL_QUADS);           //upper bench part 2
// glColor4f(0.8,0.8,0.8,0.9);
// glVertex2f(xpos+10,y14);
// glVertex2f(xpos-10,y14);
// glVertex2f(xpos+10,y14+5);
// glVertex2f(xpos-10,y14+5);
//
//
//
// glEnd();

//string A1,A2,A3,A4;
xpos=x14- cw/4;           //Top cut highwall side axis
glBegin(GL_LINES);
glColor4f(0.0,0.8,0.0,0.7);
glVertex3f(xpos,150,30);
glVertex3f(xpos,y14,30);
glEnd();
glBegin(GL_LINES);
// glColor3f(0,0,0);
glVertex3f(xpos - reach1,150,30);
glVertex3f(xpos - reach1,y7,30);
glEnd();

```



```

glBegin(GL_LINES);
//glColor3f(0,0,0);
glVertex3f(xpos,145,30);
glVertex3f(xpos- reach1,145,30);
glEnd();

glBegin(GL_QUADS);
    glColor3d(0,0,0);
    glVertex3f(xpos-15,y14,0);
    glVertex3f(xpos-15,y14+5,0);
    glVertex3f(xpos+15,y14+5,0);
    glVertex3f(xpos+15,y14,0);
    glEnd();

glBegin(GL_QUADS);
    glColor3d(1,0,0);
    glVertex3f(xpos-15,y14+15,0);
    glVertex3f(xpos+15,y14+5,0);
    glVertex3f(xpos+15,y14+15,0);
    glEnd();

glBegin(GL_TRIANGLES);
    glColor3f(0,0,0);
    glVertex3f(xpos-15,y14+15,0);
    glVertex3f(xpos+15,y14+15,0);
    glVertex3f(xpos-reach1,85,0);
    glEnd();

//here is dg code above
glBegin(GL_LINES);
//glColor3f(0,0,0);
glVertex3f(xpos-reach1,85,0);
glVertex3f(xpos-reach1,80,0);
glEnd();

glBegin(GL_LINES);
//glColor3f(0,0,0);
glVertex3f(xpos-reach1-2,77,0);
glVertex3f(xpos-reach1,80,0);
glEnd();

glBegin(GL_LINES);

```

```

//glColor3f(0,0,0);
glVertex3f(xpos-reach1+2,77,0);
glVertex3f(xpos-reach1,80,0);
glEnd();
glColor3f(1,1,1);
s="D/L-Le A1";
glRasterPos3f(xpos,150,30);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_HELVETICA_10,s[i]);
xpos=x14- 3*cw/4;           //Topcut Axis2
glBegin(GL_LINES);
glColor4f(0.0,0.8,0.0,0.7);
glVertex3f(xpos,140,30);
glVertex3f(xpos,y14,30);
glEnd();
glBegin(GL_LINES);
//glColor3f(0,0,0);
glVertex3f(xpos-reach1,140,30);
glVertex3f(xpos-reach1,y7,30);
glEnd();
glBegin(GL_LINES);
//glColor3f(0,0,0);
glVertex3f(xpos,135,30);
glVertex3f(xpos-reach1,135,30);
glEnd();
glColor3f(1,1,1);
if(mode==1){
    s="D/L-La A1";
}
else {
    s="D/L-La A2";//changed here
}
glRasterPos3f(xpos,140,30);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_HELVETICA_10,s[i]);
xpos= (x8+x16)/2;           //Axis on keycut

```

```

if(mode==1){
    s="D/L-Le A2";
    posX=reach1;
}
else if(mode==2){
    s="D/L-La A1";
    posX=reach2;
}

glBegin(GL_LINES);
glColor4f(0.0,0.6,1.0,0.7);
glVertex3f(xpos,120,60);
glVertex3f(xpos,y8,60);
glEnd();

glBegin(GL_LINES);
//glColor3f(0,0,0);
glVertex3f(xpos-posx,120,60);
glVertex3f(xpos-posx,y8,60);
glEnd();

glBegin(GL_LINES);
//glColor3f(0,0,0);
glVertex3f(xpos,115,60);
glEnd();

glColor3f(1,1,1);
glRasterPos3f(xpos,120,60);
for(int i=0;i<s.length();i++)
glBegin(GL_LINES);           //DL2 Axis rehandle
glColor4f(0.33,0.06,0.2,0.8);
glVertex3f(xd[10]+reach2,130,60);
glVertex3f(xd[10]+reach2,y7,60);
glEnd();

// Dragline 2 here
glBegin(GL_QUADS);
glColor3d(0,0,0);
glVertex3f(xd[10]+reach2-15,y7,0);
glVertex3f(xd[10]+reach2-15,y7+5,0);
glVertex3f(xd[10]+reach2+15,y7+5,0);

```

```

        glVertex3f(xd[10]+reach2+15,y7,0);
    glEnd();
    glBegin(GL_QUADS);
        glColor3d(1,0,0);
        glVertex3f(xd[10]+reach2-15,y7+15,0);
        glVertex3f(xd[10]+reach2-15,y7+5,0);
        glVertex3f(xd[10]+reach2+15,y7+5,0);
        glVertex3f(xd[10]+reach2+15,y7+15,0);
    glEnd();
    glBegin(GL_TRIANGLES);
        glColor3f(0,0,0);
        glVertex3f(xd[10]+reach2-15,y7+15,0);
        glVertex3f(xd[10]+reach2+15,y7+15,0);
        glVertex3f(xd[10],80,0);

    glEnd();
    //here is dg code above
    glBegin(GL_LINES);
    //glColor3f(0,0,0);
        glVertex3f(xd[10],80,0);
        glVertex3f(xd[10],75,0);
    glEnd();
    glBegin(GL_LINES);
    //glColor3f(0,0,0);
        glVertex3f(xd[10],75,0);
        glVertex3f(xd[10]-2,72,0);
    glEnd();
    glBegin(GL_LINES);
    //glColor3f(0,0,0);
        glVertex3f(xd[10],75,0);
        glVertex3f(xd[10]+2,72,0);
    glEnd();
    glBegin(GL_LINES);
    //glColor3f(0,0,0);
        glVertex3f(xd[10],130,60);
        glVertex3f(xd[10],y7,60);

```

```

glEnd();
glBegin(GL_LINES);
//glColor3f(0,0,0);
glVertex3f(xd[10]+reach2,125,60);
glVertex3f(xd[10],125,60);
glEnd();
glColor3f(1,1,1);
if(mode==1) s="D/L-La A2";
else if(mode==2) s="D/L-La A3";
glRasterPos3f(xd[10]+reach2,130,60);
for(int i=0;i<s.length();i++)
    glutBitmapCharacter(GLUT_BITMAP_HELVETICA_10,s[i]);
//glFlush();          //display the frame buffer
glutSwapBuffers();    //to swap the front and back buffers for animation purpose
}

void reshape(int h, int w){          //callback fn for glutReshapeFunc, called whenever the window is
    reshaped to change the clipping area so that there is no distortion of the graphics

    //viewport- the rectangular area inside the window where the graphics are displayed

    glViewport(0, 0, (GLsizei)w, (GLsizei)h);          //parameters- (origin bottom-left x of the
    viewport, y, width, height of the viewport)

    glMatrixMode(GL_PROJECTION);          //to change into projection mode

    glLoadIdentity();

    GLfloat aspect = (GLfloat)w / (GLfloat)h;

    if (w >= h) {
        // aspect >= 1, set the height from -1 to 1, with larger width

        glOrtho(-250.0 * aspect, 250.0 * aspect, -220.0, 250.0, -400, 400);
    } else {
        // aspect < 1, set the width to -1 to 1, with larger height

        glOrtho(-250.0, 250.0, -220.0 / aspect, 250.0 / aspect, -400, 400);
    }

    glMatrixMode(GL_MODELVIEW);          //to change into modelview mode. you always need to
    be inside modelview mode in order to draw
}

void init(){

```

```
    glClearColor(0.0,0.4,0.3,1); // parameters- R, G, B, alpha -- to change the background color of the
graphics window
```

```
    //glClearColor(0.37,0.2,0.63,1);
```

```
    glEnable(GL_DEPTH_TEST);
```

```
    //glOrtho(-250,250,-220,250,-400,400);
```

```
}
```

```
static void key(unsigned char key, int x, int y)
```

```
{
```

```
    switch (key)
```

```
    {
```

```
        case 32 :
```

```
        case 'q':
```

```
            exit(0);
```

```
            break;
```

```
        case 'a':
```

```
            if(angley<90)angley+=10;
```

```
            break;
```

```
        case 'd':
```

```
            if(angley>0)angley-=10;
```

```
            break;
```

```
        case 'w':
```

```
            if(anglex>0)anglex-=10;
```

```
            break;
```

```
        case 's':
```

```
            if(anglex<90)anglex+=10;
```

```
            break;
```

```
        case '1':
```

```
            mode=1;
```

```
            break;
```

```
        case '2':
```

```
            mode=2;
```

```
            break;
```

```
        case 'e':
```

```
            cw-=5;
```

```
            break;
```

```
case 'v':
    cw+=5;
    break;
case 't':
    bht+=1;
    break;
case 'r':
    bht-=1;
    break;
case 'l':
    st+=1;
    break;
case 'j':
    st-=1;
    break;
case 'Z':
    DL1prod++;
    break;
case 'X':
    DL1prod+=0.1;
    break;
case 'M':
    DL2prod++;
    break;
case 'N':
    DL2prod+=0.1;
    break;
case 'B':
    DL2prod+=0.01;
    break;
case 'z':
    DL1prod--;
    break;
case 'x':
    DL1prod-=0.1;
    break;
```

```

case 'c':
    DL1prod-=0.01;
    break;

case 'm':
    DL2prod--;
    break;
case 'n':
    DL2prod-=0.1;
    break;
case 'b':
    DL2prod-=0.01;
    break;
case 'k':
    dip-=1;
    break;
case 'i':
    dip+=1;
    break;
case '3':
    ha-=1;
    break;
case '4':
    ha+=1;
    break;
// case '1':
//     LeBoom--;
//     break;
// case '2':
//     LeBoom++;
//     break;
// case '3':
//     LaBoom--;
//     break;
// case '4':
//     LaBoom++;

```



```

        //      break;

        case 'g':

recovery-=0.01;
        break;
        case 'h':
            recovery+=0.01;
            break;
        case 'C':
            DL1prod+=0.01;
            break;
    }
    glutPostRedisplay();
}
void Specialkey(int key, int x, int y)
{
    switch(key)
    {
        case GLUT_KEY_UP:
/*Do whatever you want*/
            a+=10;
            break;
        case GLUT_KEY_DOWN:
/*Do whatever you want*/
            a-=10;
            break;
        case GLUT_KEY_LEFT:
            b-=10;
/*Do whatever you want*/
            break;
        case GLUT_KEY_RIGHT:
            b+=10;
/*Do whatever you want*/
            break;
    }
}

```

```

    glutPostRedisplay();

}

int main(int argc, char** argv){

    glutInit(&argc,argv);

    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);    //in double buffer mode of
animation, we just redraw the primitive in the back buffer and then swap it with the front buffer, and keep
on doing it. It eliminates the problem of distortion

    glutInitWindowPosition(200,00);

    glutInitWindowSize(1000,800);//window size in width x height format.

    glutCreateWindow("Balancing Diagram for Vertical Tandem");

    glutDisplayFunc(display);        //used to register a callback

    glutReshapeFunc(reshape);

    glutKeyboardFunc(key);

    glutSpecialFunc(Specialkey);

    init();

    glutMainLoop();

}

```

APPENDIX 4: LIST OF PUBLICATIONS

(A) PhD research paper in journal

1. Seervi, V., Kishore, N., & Verma, A. K. (2022). Selection of mode of tandem dragline operations by utilizing 3-dimensional computer graphics balancing diagram: a case study. Journal Of Mines, Metals & Fuels, Vol.70, No.3, 2022.
2. Seervi, V., Singh, N. P., Kishore, N., & Verma, A. K. (2022). Critical Analysis of Tandem Dragline Performance in Open Cast Coal Mines-A Case Study. Journal of The Institution of Engineers (India): Series D.

(B) PhD research paper in international conference

1. Vikram Seervi and Nawal Kishore (2021). Application of 3-Dimensional Balancing Diagram for Effective Draglines Operating in Tandem. International Conference on Opencast Mining Technology & Sustainability (ICOMS). Northern Coalfields Limited (NCL), Singrauli.